UNIVERSIDAD POLITECNICA SALESIANA

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERIA DE SISTEMAS

MENCIÓN ROBOTICA E INTELIGENCIA ARTIFICIAL

Diseño y desarrollo del hardware y software para un robot móvil con ruedas basado en sistemas de posicionamiento absoluto mediante visión artificial estereoscópica y sonar ultrasónico.

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO DE SISTEMAS

José Gabriel Egas Ortuño

DIRECTOR: Ing. Doris Bautista

Quito, abril 2012

DECLARACIÓN

Yo, José Gabriel Egas Ortuño, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

_____

José Gabriel Egas Ortuño

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Jose Gabriel Egas Ortuño bajo mi dirección.

_____

Director de tesis.

_____

Codirector.

A mi familia, por hacer posible todo lo que hago y lo que soy.

Indice

Resumen:

La robótica móvil es considerada un campo de relevante importancia para el desarrollo de la inteligencia artificial, los robots están diseñados para desenvolverse dentro de entornos muy controlados y bajo condiciones estrictas, si dichas condiciones varían, casi inevitablemente el robot fallará en la consecución de su objetivo, diseñar sistemas que estén en capacidad de enfrentar entornos desconocidos, cambiantes e inciertos, es uno de los retos mas complejos de la robótica, ofrecer soluciones a estas cuestiones significa afrontar problemas como la imposibilidad del modelamiento matemático, el tratamiento de información masiva incierta y ruidosa, manejo de incertidumbre, y sistemas de control inexactos, el presente proyecto resume muy brevemente los conceptos y generalidades relacionados con la robótica móvil, así como sus problemas y objetivos, a demás se propone el diseño y desarrollo de un prototipo de robot móvil utilizando control difuso y visión artificial estereoscópica apoyada por odometría y sonares ultrasónicos, como plataforma básica para el estudio de los problemas de la robótica móvil.

CAPÍTULO 1. ANTECEDENTES.

1.1. ANTECEDENTES HISTORICOS.

Dada la historia de la robótica, y principalmente el hecho de que las investigaciones y el desarrollo de sistemas robóticos han sido fomentados y financiados principalmente por los requerimientos de la automatización industrial, en los que el primer problema a resolver es la automatización de los procesos de manufactura; el estudio, la técnica y la evolución de los robots manipuladores han sufrido un impulso considerablemente mayor al impulso dado al desarrollo de robots móviles; al ser los procesos de manufactura de carácter repetitivo y considerándose los requerimientos de los sistemas industriales de fabricación y producción en masa, los robots manipuladores industriales no han tenido la necesidad de enfrentarse a ambientes cambiantes, o simplemente el ambiente se ha controlado artificialmente con la finalidad de posibilitar el desenvolvimiento del robot, así, un manipulador que cumple una determinada tarea dentro de un proceso de fabricación esta diseñado o programado únicamente para cumplir esta tarea específica, es decir, está diseñado para enfrentarse a un entorno específico en el que, de manera general, la posición, velocidad y aceleración de cada objeto, así como la posición de cada uno de los eslabones de la cadena cinemática que conforma el manipulador, es controlada estrictamente, esto facilita enormemente el desarrollo del software que controla al robot, ya que, de cierta manera, los procesos sensoriales, perceptivos e inteligentes son realizados por el programador y no por el sistema en sí, si bien ha sido este hecho el que ha posibilitado el desarrollo y la aplicación de la robótica en la industria, actualmente la capacidad computacional ofrece la posibilidad de desarrollar sistemas sensoriales y perceptuales que posibiliten a las máquinas el afrontar entornos dinámicamente cambiantes, tomar decisiones frente a situaciones imprevistas y alcanzar objetivos claros mediante su actuar.

1.2. IMPORTANCIA DE LA VISIÓN PARA LA MOVILIDAD.

Podemos afirmar con gran certeza que la captura y procesamiento de imágenes es el proceso mediante el cuál se puede obtener de la manera mas eficiente y rápida gran cantidad de información imposible de obtener mediante otros métodos y de relevante importancia para un agente que se enfrenta a un entorno desconocido, sea éste biológico o artificial, la naturaleza ha demostrado la fundamental importancia de la visión para la movilidad, si bien muchos organismos basan su movimiento en otros tipos de sistemas sensoriales, especialmente aquellos para quienes la luz es muy escasa en su entorno, los organismos mas evolucionados, los mamíferos, usan casi en su totalidad la visión como método para obtener información de su entorno, es evidente que los organismos biológicos enfrentan condiciones de entorno mucho mas complejas que los robots, los sistemas perceptuales que se utilizan para manejar la masiva cantidad de información entregada por las imágenes capturadas por los ojos posibilitan al agente a enfrentarse con una gran variedad de escenarios distintos, reaccionar de manera eficiente ante los cambios del entorno y procurar su subsistencia.

1.3. IMPORTANCIA DE LA VISIÓN ESTEREOSCÓPICA PARA LA MOVILIDAD.

La estereoscopía permite, a partir de dos imágenes bidimensionales capturadas desde posiciones distintas, construir una imagen tridimensional, requerimiento importante para la movilidad basada en la visión, mediante estereoscopía es posible obtener la posición tridimensional de un objeto que se encuentre en el campo visual estereoscópico, en tanto si se utiliza visión mono-ocular o si el objeto está fuera del campo visual estereoscópico, se podrá obtener únicamente un posicionamiento bidimensional, o recurrir a la estereoscopía mono-ocular, falencia que se

intenta suplir, en los animales dotados de visión mono-ocular, mediante procesos perceptuales menos eficientes como la percepción de tamaño, forma, velocidades, etc.

1.4. PROBLEMAS RELACIONADOS CON LA VISIÓN.

Es de notar que, en absolutamente todos los agentes biológicos que basan su movilidad en la visión, sea esta mono o binocular, se encuentra una gran porción del cerebro dedicada al procesamiento de las imágenes, en muchos casos, como en los humanos, aproximadamente dos tercios de la corteza cerebral están relacionados directa o indirectamente con la visión, situación ésta que da un indicio de la complejidad de los algoritmos necesarios para el tratamiento de las imágenes, de dicha complejidad se deriva el principal problema que enfrentan los sistemas de visión artificial, a saber, que el tiempo de ejecución de un algoritmo de procesamiento en una imagen relativamente grande es muy largo, imposibilitando en muchos casos, que las reacciones del robot puedan ser consideradas en tiempo real; si bien el incremento de la capacidad computacional de los procesadores ha abierto la posibilidad a la utilización de muchos algoritmos de visión artificial, muchos otros, como los de reconocimiento de patrones, siguen tomando mucho tiempo en su ejecución si se los quiere aplicar a ciertas circunstancias que requieren de reacciones inmediatas. La adaptabilidad de los sensores visuales es otro de los principales problemas relacionados con la visión artificial, comprendemos la adaptabilidad como la capacidad de cambiar parámetros del sensor frente a distintos ambientes, por ejemplo la capacidad de cambiar la distancia focal y variar el enfoque dependiendo de la distancia a la que un objeto se encuentre, la capacidad de variar la cantidad de luz que ingresa al sensor dependiendo de la luminosidad del ambiente, así como las características de dichas capacidades, principalmente el tiempo de reacción. En el caso de las cámaras usadas en sistemas robóticos estas capacidades, si es que

existen, son enormemente limitadas si se las compara con las capacidades de los ojos.

1.5. PROBLEMAS DE LA ROBÓTICA MÓVIL, EL PROBLEMA SLAM.

El problema principal de la robótica móvil es la determinación de la posición y orientación de un robot dentro de su entorno, el problema denominado SLAM (Localización y Mapeo Simultáneos, Simultaneous Localizarion And Mapping) ha sido planteado y estudiado desde hace aproximadamente 20 años, un sistema SLAM debe considerar principalmente los siguientes obstáculos en su diseño:

- Los sensores utilizados en un robot móvil, como los sonares ultrasónicos, las cámaras digitales, los odómetros y los escaners láser, están sujetos a ruido y perturbaciones a su funcionamiento, por lo tanto el algoritmo SLAM debe estar en capacidad de interpretar y manejarse con mediciones inexactas, es por este motivo que la determinación exacta de la posición de un robot móvil en un entorno no modificado para su funcionamiento, resulta siempre imposible, es decir, siempre existirá un error, ya sea este acumulable o no, en la posición hallada.
- Los algoritmos SLAM mas exitosos planteados hasta la fecha, se basan en la ESTIMACION y no pretenden la DETERMINACION de la posición del robot, cabe mencionar que dichos algoritmos usan principalmente estimaciones probabilísticas, lógica difusa, y otros métodos inexactos, mas no modelos matemáticos exactos.
- La generación de MAPAS del entorno se hace, dentro del problema SLAM esencial, sin modificar el entorno del robot, de esta manera el

robot debe enfrentarse con un entorno desconocido, complicándose significativamente el problema.

CAPÍTULO 2. MARCO TEÓRICO.

A continuación se muestra una breve introducción a los conceptos básicos sobre los que se fundamenta el trabajo posterior.

2.1. MÉTODOS DE POSICIONAMIENTO DE ROBOTS MÓVILES.

Se entiende como un método de posicionamiento a aquel que permite al robot estimar su posición (después de realizar uno a varios movimientos) relativa a un sistema de coordenadas dado, este sistema de coordenadas es arbitrario y puede ubicarse, por ejemplo, en el punto en el que se inician los movimientos, sobre una marca del entorno o en un punto de referencia dado mediante software, estos métodos de control se dividen en dos grandes grupos:

2.1.1. SISTEMAS DE POSICIONAMIENTO ABSOLUTO.

Se comprende como un método de posicionamiento absoluto a aquel que utiliza señales del entorno de desenvolvimiento del robot, para la obtención de la posición actual y la posterior consecución de la posición deseada, dichas señales del entorno, a su vez, se clasifican en dos:

2.1.1.1. Marcas o señales artificiales.

Son aquellas que se han agregado intencionalmente al entorno con la finalidad de proporcionar información al robot, entre estas podemos mencionar al GPS, emisores infrarrojos o ultrasónicos, líneas o manchas agregadas al piso, paredes, etc.

2.1.1.2. Marcas o señales naturales.

La utilización de marcas no agregadas intencionalmente al entorno, como bordes, esquinas, paredes, objetos como sillas, lámparas, u otros componentes ¨naturales¨ del entorno de movimiento del robot, supone el desarrollo de algoritmos inteligentes capaces del tratamiento de información incierta, masiva y variante, dichos algoritmos constituyen un tema de investigación actual frente al cuál no se han dado respuestas definitivas, centralizándose el problema en el desarrollo de algoritmos de visión artificial.

2.1.2. SISTEMAS DE POSICIONAMIENTO RELATIVO.

Se comprende como un sistema de posicionamiento relativo a la obtención de información concerniente a la posición del robot a partir de sus propias características, es decir las que pertenecen al cuerpo del agente, diferenciándolo de su entorno, dichas características pueden incluir:

2.1.2.1. Características inerciales de los movimientos.

Para su obtención se utilizan sensores de aceleración, con la finalidad de poder estimar el movimiento a partir de

mediciones de los momentos de inercia, fuerzas generadas con la aceleración lineal o angular, como las centrífugas y de Coriolis, utilizando giróscopos, acelerómetros, etc.

2.1.2.2. Características geométricas y mecánicas del vehículo.

La obtención de éstas supone el conocimiento previo del modelo y diseño del vehículo, en esta información se incluye el diámetro de las ruedas, velocidad, sentido y ángulo de giro de los motores, etc. Y, dependiendo de la configuración de las ruedas utilizadas, es posible estimar los desplazamientos realizados por el robot. Para la obtención de la velocidad, sentido y ángulo de giro de los motores es común el empleo de sistemas odométricos mediante potenciómetros o encoders, siendo estos últimos los más utilizados y difundidos por presentar mejores características y ciertas ventajas frente a los potenciómetros, de todas maneras, y a pesar de que se diseñe un sistema con sensores redundantes (encoder y potenciómetro), la acumulación del error durante las mediciones repetitivas y al ser éstas miles o cientos de miles en un solo movimiento del robot, así como la suposición errónea de que un movimiento del motor se traduce siempre en un movimiento del robot, sin considerar deslizamientos no previstos entre la rueda y el piso, determina que sea imposible obtener una posición exacta después de realizar una serie de giros de las ruedas, de esta manera, las técnicas de odometría se utilizan como sistemas de apoyo o complementarios a los sistemas de marcas, naturales o artificiales, en el entorno.

2.1.2.3. Odometría, problemas.

Un sistema de codificador óptico es un sensor de error acumulable, es decir, el error producido en cada medición se suma al error de la medición siguiente, de esta manera cuantas mas mediciones se realice, cuanto mayor sea el ángulo de giro del codificador óptico, mayor será el error en la estimación de la posición, este es el motivo fundamental que hace imposible el cálculo de la posición final del robot después de una secuencia de movimientos, aún si se garantiza el hecho de que el coeficiente de rozamiento entre las ruedas del móvil y la superficie es muy alto y, para fines prácticos, pueda considerarse infinito, la acumulación progresiva del error hace que solo sea posible hacer una estimación de la posición final, por esta razón los sistemas de odometría son utilizados en robótica móvil en conjunto con otros mecanismos de posicionamiento, de preferencia absolutos, para la estimación de la posición, entre las causas mas comunes e importantes que añaden error a los sistemas odométricos están:

- Errores en los datos tomados de la estructura del robot:

  Cuando se realiza el modelamiento matemático que predice la descripción de un movimiento, sea éste un giro o un desplazamiento, el conjunto de ecuaciones resultantes inevitablemente tendrá como variables independientes parámetros de la estructura física del robot, tales como el diámetro de las ruedas, la relación de transmisión de las cajas reductoras de velocidad que se acoplan a los motores con los ejes de las

ruedas, la posición relativa de una rueda con respecto a la otra, y ciertas dimensiones físicas de la estructura del móvil, si dichos valores no han sido obtenidos con exactitud, o si varían, por ejemplo, con el desgaste de las piezas, estaremos introduciendo mayor error en el cálculo de la posición final mediante las ecuaciones de odometría.

Sin embargo, exceptuando el caso de magnitudes que puedan cambiar con el tiempo, como son el diámetro de las ruedas debido al desgaste o a la presión de aire en su interior, el error introducido por las medidas erradas es constante, es decir, producirá siempre el mismo porcentaje de error en la posición final obtenida, se han logrado buenos resultados determinando experimentalmente este porcentaje y haciendo una corrección mediante software, por ejemplo, si se pretende hacer un desplazamiento recto de distancia X y se obtiene constantemente, debido a los errores de medición, una distancia total de $X+nX/100$, es posible estimar experimentalmente el valor de n y restarle el porcentaje $nX/100$ al valor de entrada X , así, si se pretende un desplazamiento de longitud X o un giro de A grados, lo que realmente se ejecutará será un desplazamiento de $X-nX/100$ o un giro de $A+-nA/100$ según sea el caso, con esto evitamos el tomar mediciones muy exactas y la necesidad de instrumentos de medida muy refinados.

- Errores externos, producidos por factores impredecibles:

En general, todos los factores externos que puedan producir un deslizamiento no previsto entre la rueda y su superficie de contacto, por ejemplo bajo coeficiente de fricción, aceleraciones o desaceleraciones bruscas, desniveles imprevistos en el piso, objetos que quedan bajo las ruedas, fluidos lubricantes derramados, etc.

Es importante acotar que el carácter impredecible de este tipo de factores hace que sea imposible modelarlos, sin embargo, el error en la posición introducido por los factores externos se elimina por completo una vez que el robot tiene a su disposición una marca externa, sea ésta de carácter natural o artificial, que permite determinar su posición, es por esta razón que los sistemas de posicionamiento absoluto se tornan inevitables en un robot móvil.

2.2. REPRESENTACIÓN Y CARACTERÍSTICAS DE LOS ROBOTS MÓVILES CON RUEDAS.

2.2.1. CLASIFICACIÓN DE LAS RUEDAS.

2.2.1.1. Rueda no motriz.

Una rueda no motriz es aquella que no ejerce un torque sobre el piso, se representará mediante un rectángulo vacío:

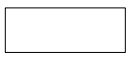

Figura 1. Rueda no motriz.
Fuente: El autor.

2.2.1.2. Rueda de castor.

Constituye una rueda no motriz orientable de eje descentrado, se la puede considerar únicamente como un punto de apoyo que no ejerce fuerza alguna sobre la estructura del robot.

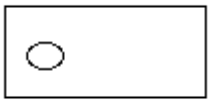

Figura 2. Rueda de castor.
Fuente: El autor.

2.2.1.3. Rueda motriz.

Una rueda motriz es aquella que ejerce un torque sobre el piso, sea éste para detener o impulsar al robot, se representará mediante un rectángulo relleno o coloreado de negro:

Figura 3. Rueda motriz.
Fuente: El autor.

2.2.1.4. Rueda direccional.

Es aquella que puede variar su ángulo de inclinación y modificar de esta manera la dirección del vehículo, se representa mediante una rueda, motriz o no, encerrada en un

círculo:

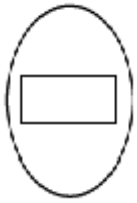

Figura 4. Rueda no motriz direccional.
Fuente: El autor.

Figura 5. Rueda motriz direccional.
Fuente: El autor.

2.2.1.5. Oruga.

La oruga es considerada y analizada, en la mayoría de textos de robótica móvil, como una rueda, aunque es oportuno aclarar que, debido principalmente a la superficie de contacto con el piso, la oruga posee características que no se hallan en una rueda, por lo general es motriz y no direccional, aunque se han usado motrices direccionales y no motrices direccionales.

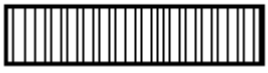

Figura 6. Oruga.
Fuente: El autor.

2.2.1.6. Rueda omnidireccional.

Una rueda omnidireccional puede ser descrita como una rueda a la que se la han añadido ruedas, consiste de una rueda con un eje central a la que se han acoplado, alrededor de su perímetro, un conjunto de cilindros con ejes de rodamiento paralelos entre sí, pero que forman un determinado ángulo con el eje de la rueda central, dichos cilindros permanecen en contacto con el suelo, de esta manera una rueda omnidireccional puede desplazarse independientemente de su dirección, y, en conjunto con las fuerzas ejercidas por otras ruedas de la misma clase, ejercer un torque sobre el suelo.

Figura 7. Rueda omnidireccional.
Fuente: El autor.

2.3. MOVIMIENTOS, CENTRO INSTANTÁNEO DE ROTACIÓN Y CENTRO DE GUIADO.

Un vehículo con ruedas puede realizar dos tipos de movimientos: giros y desplazamientos, se considera un giro a todo movimiento que permita identificar un punto dentro del volumen ocupado por el robot que no se desplace en absoluto, a dicho punto se lo llama centro de guiado (Cg), un desplazamiento no permite identificar el centro de guiado, si éste existe en la configuración de ruedas a estudiar, se puede hallar mediante un giro, y debe ser el mismo tanto en giros

horarios como anti horarios.

El centro instantáneo de rotación o CIR es el punto de intersección de las prolongaciones de los ejes de todas las ruedas con excepción de las de castor, al desplazarse el vehículo su Cg describirá una trayectoria circular con centro en el CIR, un caso especial lo constituye el robot diferencial, en el que el CIR esta dado en función de la diferencia de velocidad de las dos ruedas motrices, si los ejes no intersecan, o si intersecan en infinitos puntos, el robot avanza en línea recta.

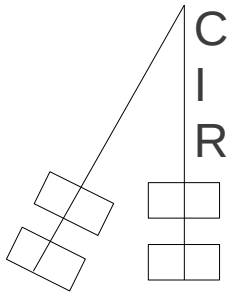

Figura 8. Centro instantáneo de rotación en un vehículo Ackerman.
Fuente: El autor.

2.3.1. MODELAMIENTO EN EXISTENCIA DEL CENTRO DE GUIADO.

De manera general se puede afirmar que el modelamiento matemático de un robot móvil se facilita si la configuración de ruedas de dicho robot permite la identificación de un centro de guiado, si esta condición se da es posible considerar al vehículo como una partícula adimensional y aplicar por ejemplo las técnicas de expansión de obstáculos para el modelado del entorno, también es posible analizar la cinemática del vehículo considerándolo como una magnitud vectorial en la que el origen está en el centro de guiado, la

magnitud y el sentido resultan de la suma vectorial de las fuerzas aplicadas por las ruedas, lo cuál permite la obtención de un modelo predictivo de los movimientos.

2.3.2. IMPORTANCIA SENSORIAL DEL CENTRO DE GUIADO.

El centro de guiado constituye un punto de fundamental importancia en el posicionamiento de los sensores a utilizar, en el caso de cámaras o sensores ultrasónicos de distancia, si se ubican en el centro de guiado se facilitará el manejo de la información obtenida, posibilitando una congruencia entre el modelo matemático del robot y los datos sensoriales y perceptivos, esto se deriva del hecho de que las mediciones han sido hechas desde un punto perfectamente conocido dentro del espacio corporal del vehículo.

2.4. CLASIFICACIÓN DE LOS ROBOTS CON RUEDAS.

La clasificación de los robots móviles se basa en su morfología, es decir en la forma de su estructura física, específicamente se clasifican en función de la disposición espacial de las ruedas que se utilicen, se presentan a continuación las principales arquitecturas, una breve descripción de sus características y su correspondiente representación gráfica:

2.4.1. CONFIGURACIÓN TRICICLO.

> El robot esta formado por mínimo tres ruedas, una de ellas es motriz direccional, las restantes son no motrices y no direccionales.
> (Ollero,2001,29)

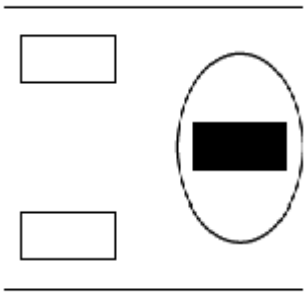

Figura 9. Configuración de triciclo.
Fuente: El autor.

La ventaja de esta configuración radica en el hecho de que existe una completa independencia entre motricidad y direccionamiento, simplificándose significativamente tanto el modelamiento matemático como la programación del robot, la principal desventaja radica en la imposibilidad de realizar giros sobre si mismo, es decir, en esta configuración no existe un centro de guiado.

2.4.2. CONFIGURACIÓN DIFERENCIAL.

Consta de dos ruedas motrices no direccionales situadas diametralmente sobre un mismo eje perpendicular al frente del robot, al menos una rueda loca como punto de equilibrio, la dirección del robot varia en función de la diferencia de velocidad entre la rueda izquierda y derecha, si la velocidad es la misma el robot avanzará describiendo una recta, presenta la ventaja de un centro de guiado en la mitad del eje perpendicular que pasa por los centros de las ruedas motrices.
(Bräunl,2006,98)

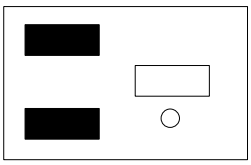

Figura 10. Configuración diferencial.
Fuente: El autor.

2.4.3. CONFIGURACIÓN ACKERMAN.

Se consigue mediante dos ruedas direccionales paralelas anteriores y dos ruedas motrices posteriores, no es una configuración frecuentemente usada en robots móviles debido a su escasa maniobrabilidad y a su complejidad de construcción si se la compara con las anteriores.
(Bräunl,2006,105)

2.5. MÉTODOS DE MODELAMIENTO DEL AGENTE - ENTORNO, FUNCIONES Y ESPACIOS.

2.5.1. FUNCIÓN DE GIROS Y DESPLAZAMIENTOS.

Es una o varias funciones matemáticas cuya variable de salida es la posición relativa (X,Y,ángulo) del robot dentro de su plano de movimiento y cuyas variables de entrada son los ángulos de las secuencias de movimientos realizados por las ruedas, es decir, permite estimar y predecir la posición del vehículo dados ciertos movimientos angulares de las ruedas, se da por hecho que el coeficiente de rozamiento entre las ruedas y el piso es alto y que no

habrán deslizamientos, la obtención de esta función o conjunto de funciones se simplifica mucho en el caso de existir un centro de guiado en la configuración a utilizar, y se logra en la mayoría de los casos mediante un simple análisis geométrico de la estructura del robot y la aplicación de ecuaciones trigonométricas.

2.5.2. MAPAS POLIGONALES Y POLIGONALES PARAMÉTRICOS.

Comprenderemos a un mapa como un plano cartesiano en el que se definen polígonos regulares cerrados mediante conjuntos de pares ordenados que representan los vértices de cada polígono, el área dentro de cada polígono se considera un área ocupada y por lo tanto no disponible para desplazamientos del robot.

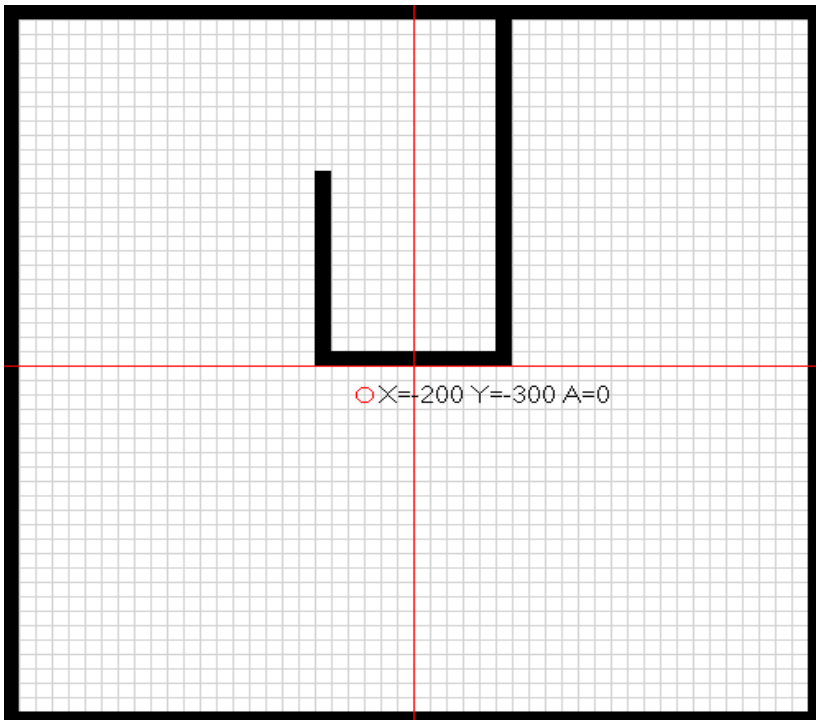

Figura 11. Un ejemplo de mapa poligonal.
Fuente: El autor.

En la figura anterior, podemos apreciar un mapa poligonal que representa un área cerrada por paredes, la posición estimada del robot se simboliza mediante el círculo, a continuación se muestran, usando C#, los pares ordenados que han definido a uno de los polígonos regulares cerrados que representa una de las paredes interiores del área de desplazamiento:

```
PointF[] o1=new PointF[4];
o1[0]=new PointF (0,0);
o1[1]=new PointF (10,0);
o1[2]=new PointF (10,250);
o1[3]=new PointF (0,250);
```

Como se ve, cada par ordenado es un vértice del polígono.

Dependiendo de las propiedades que se requieran, se puede definir el origen de coordenadas del mapa sobre el centro de guiado del robot o sobre un punto conveniente como puede ser el centro del mapa.

2.5.3. ESPACIO CORPORAL.

Se define al espacio corporal como un modelo de sí mismo, es una representación que el robot tiene de su propia estructura y dimensiones, el modelamiento del espacio corporal se puede evitar o no dependiendo del método de modelamiento del entorno que se realice, es recomendable modelar el espacio corporal como una función matemática o un mapa poligonal de un polígono regular cerrado que abarque toda la estructura del robot, aunque también puede hacerse mediante una imagen como un mapa de bits, es importante la programación de un modelo de espacio corporal para evitar que el robot intente ingresar por lugares estrechos, en los que su cuerpo no cabrá, así como evitar choques con objetos del entorno

que están alejados del centro de guiado y dentro del espacio corporal, es muy común el modelamiento del espacio corporal usando un polígono simple semejante al contorno del robot, como un rectángulo o un círculo, es importante también considerar la posición del centro de guiado dentro del espacio corporal.

2.5.4. ESPACIO ALOCÉNTRICO.

El espacio alocéntrico es una representación “alo, distinto”, que en robótica se podría definir como un mapa poligonal, mapa poligonal paramétrico o un mapa de pixeles del entorno conocido del robot. Ofrece una “vista global” del entorno, una perspectiva independiente del observador, en la que se ubican los espacios ocupados y los espacios libres para posibles desplazamientos, si el mapa alocéntrico es dado por el programador al robot, se dice que el robot tiene un conocimiento previo del entorno, se simplifican de esta manera los procesos perceptivos y se limita la inteligencia del robot, en cambio, si se pretende que el robot se enfrente a entornos desconocidos, son necesarios procesos de percepción, principalmente percepción visual, que permitan la generación automática de un mapa alocéntrico previa o durante la ejecución de los movimientos.

2.5.5. ESPACIO EGOCÉNTRICO.

Constituye una representación del conjunto agente – entorno, al igual que el espacio alocéntrico puede ser representado a manera de mapa poligonal, poligonal paramétrico o mediante una imagen. Tiene como objetivo brindar una perspectiva de la ubicación de los estímulos que provienen de los objetos del entorno tomando como punto de referencia las coordenadas de la posición del agente, es decir, ubica al robot en la posición X=0,Y=0 y a cada objeto del

entorno se le asigna un volumen de ocupación dentro de unas coordenadas que toman como referencia al robot. De ser posible la ubicación del eje de coordenadas coincidirá con el centro de guiado, esto facilita el análisis.

2.5.6. REJILLAS DE OCUPACIÓN.

El método de rejillas de ocupación es un sistema utilizado para simplificar las características del entorno dentro de un espacio o mapa que lo representa, en una de sus formas básicas, este método modela la superficie sobre la que se desplazará el robot a manera de una rejilla, de la que cada campo puede tener dos posibles estados: ocupado o libre, los campos ocupados representan objetos del entorno que imposibilitan el desplazamiento del robot, como paredes, gradas, mesas, sillas, etc. En tanto que los campos libres representan espacios disponibles para ser ocupados sin posibilidad de choque, un mapa por rejillas de ocupación puede considerarse como un mapa poligonal, es decir, pueden determinarse las posiciones tamaños y formas de los objetos del entorno mediante polígonos que cubran mínimo una rejilla, estos polígonos suelen determinarse mediante coordenadas absolutas o paramétricas, y es posible localizar el origen de coordenadas en cualquier punto de conveniencia dentro del mapa, usualmente este origen coincide con la posición inicial de partida del robot.

2.5.7. EXPANSIÓN DE OBSTÁCULOS.

La expansión de obstáculos es un método utilizado para evitar la complicación que se presenta al momento de modelar el espacio corporal del robot, puesto que el espacio corporal es función de la posición del robot mientras realiza desplazamientos o giros, es decir, las coordenadas de los puntos que representan la estructura física o

el cuerpo en si del robot, varían mientras éste se mueve, es preferible expandir el perímetro de todos los obstáculos (las rejillas ocupadas) una dimensión que corresponde con la distancia entre el punto mas alejado de la estructura del móvil y su centro de guiado, así, el vehículo puede ser considerado como una partícula que se desplaza dentro de su entorno, evitándose la necesidad del desarrollo de un modelo o mapa de espacio corporal.

2.6. SENSORES ULTRASÓNICOS DE TIEMPO DE VUELO.

Un sensor ultrasónico de tiempo de vuelo basa su funcionamiento en el hecho de que el sonido tiene una velocidad de desplazamiento aproximadamente constante en el aire, aunque dicha velocidad varia en función de ciertos parámetros, como son la humedad y la presión atmosférica, dichas variaciones suelen ser mínimas y no se consideran en los sensores ultrasónicos, el sensor emite un pulso ultrasónico e inmediatamente inicia a contar el tiempo, se detiene una vez superado un límite o cuando se recepta un eco, si el eco existe, significa que el sonido ha rebotado sobre un objeto dentro del alcance de medida del sensor, y el tiempo representa el tiempo de vuelo del sonido, con la simple ecuación de velocidad (v=e/t) y dado que v es constante y se conoce el tiempo, es posible calcular e, que en este caso representa la distancia o el espacio hasta el objeto.

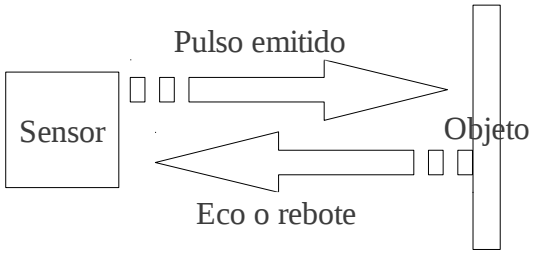

Figura 12. Un objeto detectado.
Fuente: El autor.

2.6.1. PROBLEMAS DE LOS SENSORES ULTRASÓNICOS.

Si bien el funcionamiento del sensor ultrasónico es relativamente simple, éstos están sujetos a importantes limitaciones que se deben considerar en su uso, se describen a continuación brevemente dichas limitaciones:

2.6.1.1. Ángulo de rebote.

El sensor detectará un obstáculo únicamente si el ángulo de rebote del sonido está dentro de sus límites de detección, es decir, si la cara de un objeto supera el ángulo límite, el sensor no recibirá el eco y no detectará al objeto o lo detectará a una distancia errada:

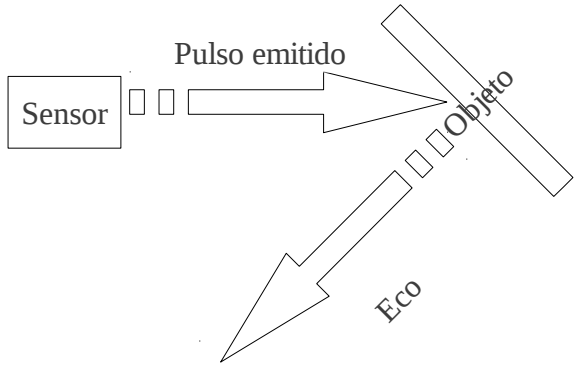

Figura 13. Un objeto que no se detecta.
Fuente: El autor.

En la figura anterior se ilustra el caso de un objeto que no se detecta puesto que el ángulo está fuera del límite de detección del sensor.

2.6.1.2. Falsos rebotes.

Este caso suele ocurrir en esquinas o lugares cerrados compuestos por varias paredes, sucede cuando un pulso de sonido rebota varias veces antes de regresar al sensor, en este caso pueden detectarse distancias mayores a las reales.

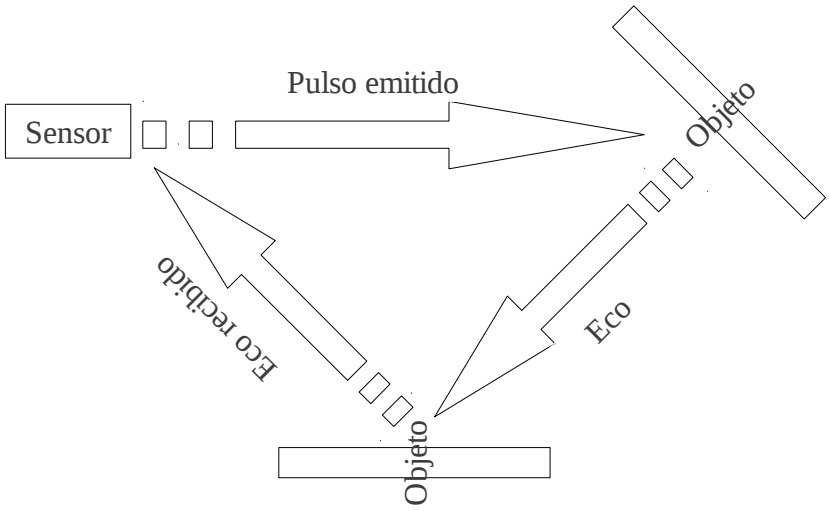

Figura 14. Un error en la medida producido por un rebote múltiple.
Fuente: El autor.

En el ejemplo de la figura 14, el objeto se detecta, pero puesto que la distancia recorrida por el pulso ultrasónico es mayor, se detectará a una distancia mayor a la real.

2.6.1.3. Esquinas y aristas.

Si el sensor se ubica frente a una esquina o una arista, ésta no se detectará o se detectará a una distancia mucho mayor a la real, en este caso la magnitud leída depende de el ángulo de la arista.

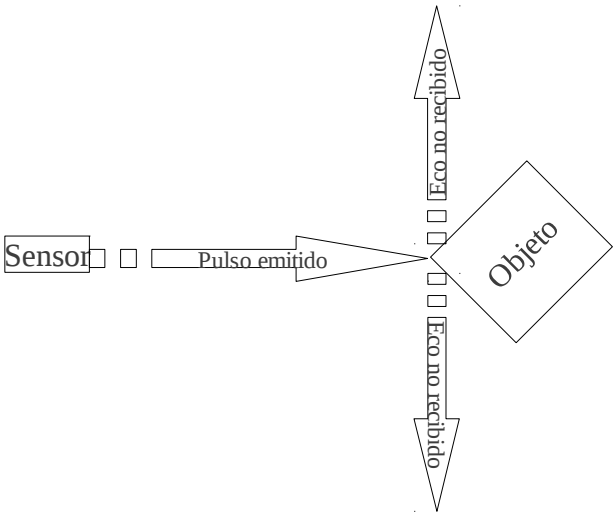

Figura 15. Una esquina que no se detecta.
Fuente: El autor.

2.6.1.4. Uso de varios sensores.

El uso de varios sensores dentro de un mismo ambiente es con seguridad la fuente mas frecuente de mediciones erradas y produce también ¨objetos fantasma¨, cuando el pulso de un sensor es recibido por otro, éste detectará un objeto inexistente, o simplemente dará una medida completamente arbitrarea.

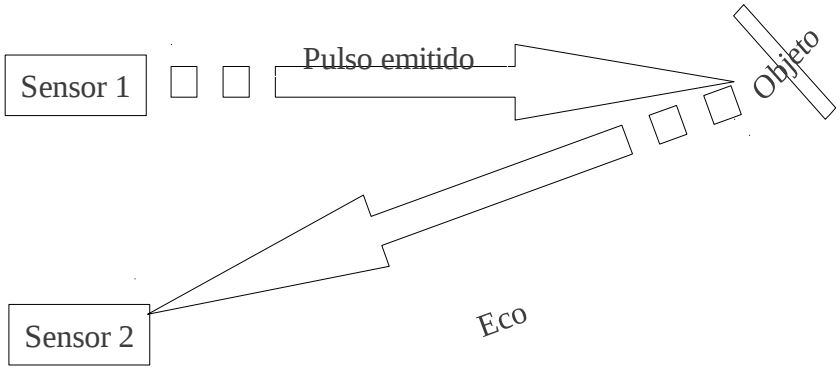

Figura 16. Una falsa medida y un ¨objeto fantasma¨ producido por el uso simultáneo de dos sensores ultrasónicos.
Fuente: El autor.

2.6.1.5. Objetos de materiales absorbentes.

Ciertos materiales blandos pueden absorber el pulso ultrasónico emitido por el sensor, o reflejar un pulso de intensidad mucho menor, en este caso un objeto absorbente puede no detectarse.

2.6.1.6. Objetos con superficies rugosas o disformes.

En ocasiones este tipo de objetos pueden dispersar el sonido, no siendo detectados.

2.6.2. ANILLO DE ULTRASONIDOS.

Puesto que los sensores ultrasónicos presentan gran facilidad de uso, y a pesar de los varios inconvenientes y la incertidumbre que de éstos se deriva, se suele construir al rededor del robot móvil un arreglo a manera de ¨anillo¨ dispuesto de un número, generalmente potencia de 2, de sensores ultrasónicos, todos apuntando en distintas direcciones y, por lo general, priorizando las lecturas al frente, en segunda instancia a los lados y por último detrás del robot.

2.6.3. MÉTODOS PARA MANEJAR LA INCERTIDUMBRE.

Con al finalidad de evitar que el robot tome acciones basadas en medidas erradas, se trata la incertidumbre en las lecturas mediante diversos métodos, entre los cuales:

2.6.3.1. Filtrado mediante software.

Se puede filtrar los datos errados tomando en cuenta la velocidad máxima de desplazamiento del robot o si éste se

desplaza o no con respecto a un objeto, como una pared, por ejemplo, si se mide la distancia a un objeto y se encuentra un valor de 35 cm, en la siguiente medida se lee un valor de 600 cm, y en los posteriores se lee nuevamente 35 cm, es de suponer que, si las medidas se tomaron por ejemplo en intervalos de 100 milisegundos, es imposible que dicho objeto se haya alejado de 35 cm a 600 cm y luego regresado nuevamente a los 35 en un período igual o menor a 200 milisegundos, o que exista una grieta en la pared que de este patrón de lecturas, en consecuencia, se diseña un simple algoritmo que filtre este dato erróneo.

2.6.3.2. Control de la emisión de pulsos ultrasónicos.

Con la finalidad de evitar la aparición de ¨objetos fantasma¨ y de múltiples rebotes de pulsos ultrasónicos entre varios sensores, es recomendable controlar y sincronizar la emisión de pulsos por cada sensor, los sensores comúnmente usados como el SRF05 poseen un pin que inicia o detiene la emisión de ultrasonido, en este caso se puede usar un microcontrolador que active al siguiente sensor desactivando al previo, así podemos asegurarnos que cada sensor toma una medida a la vez, y no se emiten dos pulsos simultáneos, un sistema algo mas complejo podría priorizar las medidas haciéndolas mas frecuentes en los sensores que adquieren mas importancia en el diseño del robot, pero, de igual manera, evitar que dos sensores emitan pulsos a la vez, otros sensores, como el LVEZ4, poseen ya un sistema de conexión en ¨cascada¨ que permite a cada sensor comunicarse con el siguiente del anillo para avisarle que ha terminado su proceso de medida y así se sincronizan entre ellos.

2.7. CONTROL DIFUSO.

El control difuso es el control de una variable realizado mediante lógica difusa o multivaluada, esencialmente la lógica difusa maneja n valores de verdad entre 0 (totalmente falso) y 1 (totalmente verdadero) de tal manera que una proposición puede tomar cualquier valor de verdad entre 0 y 1, por ejemplo 0.6. Los conjuntos de objetos dentro de lógica difusa son similares a los de la teoría de conjuntos de la lógica clásica o binaria, y se aplican también operaciones como unión, intersección, diferencia, subconjuntos, etc.

2.7.1. VENTAJAS.

La principal ventaja del uso del control difuso es que permite el modelamiento de sistemas que son en extremo difíciles de modelar matemáticamente, o que simplemente es imposible obtener un modelo matemático exacto, la lógica difusa adquiere especial importancia dentro de la robótica móvil puesto que el modelamiento matemático de muchos de los aspectos de un móvil es muy complejo y que la información que se debe tratar es incierta y ruidosa.

2.8. VISIÓN ESTEREOSCÓPICA.

2.8.1. VISIÓN Y ESTEREOSCOPÍA MONO-OCULAR.

Si bien la visión mono-ocular puede parecer suficiente para el caso de un robot móvil que desea desplazarse por un ambiente desconocido, ésta no posee la característica fundamental de entregar información con respecto a la profundidad a la que se encuentra determinado objeto sin el conocimiento previo de ciertas características del mismo, como su tamaño, color y textura, etc. Existen mecanismos perceptuales que permiten, mediante un sistema mono-ocular, estimar la distancia hacia determinado objeto

en la imagen, el inconveniente de dichos mecanismos es que basan su funcionamiento en experiencias y conocimientos previos, por ejemplo, es posible estimar la profundidad mediante:

- El tamaño relativo del objeto en cuestión. Se estima la profundidad mediante el tamaño (área) del objeto en la imagen bidimensional, puesto que mientras mas lejos se encuentra un objeto menor tamaño relativo tendrá.
- Es posible estimar la profundidad mediante el grado de detalle o nitidez de la imagen obtenida, menor detalle y nitidez significa mayor distancia.
- La velocidad relativa aparente de desplazamiento de un objeto; mientras mayor sea la distancia entre un objeto y un observador menor parecerá la velocidad relativa del objeto al moverse el observador.
- Perspectiva, es posible estimar la profundidad mediante las reglas geométricas que definen la perspectiva.
- Enfoque, el enfoque (distancia focal que determina una imagen clara) varía según la distancia, para usar este método se requiere el conocimiento previo de parámetros físicos de la cámara usada así como la posibilidad de variar la distancia focal.

2.8.2. COMPONENTES DE UN SISTEMA ÓPTICO.

Se presentan a continuación los principales componentes y características de un sistema óptico basado en cámaras:

2.8.2.1. Ejes ópticos y plano de la imagen.

Se comprende como eje óptico a la línea imaginaria perpendicular al plano de la imagen, en el caso de una cámara el eje óptico atraviesa perpendicularmente los centros de los componentes del sistema óptico así como el centro exacto del plano de la imagen, el plano de la imagen es la imagen bidimensional en si, y es paralelo al sensor o

matriz sensorial que capta la imagen.

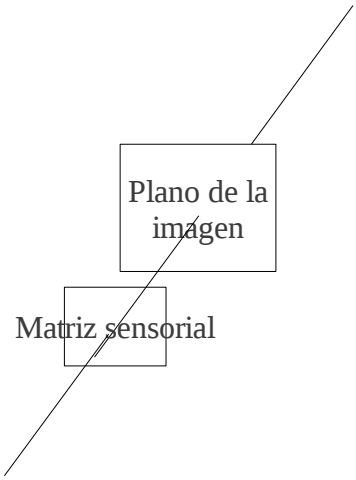

Figura 17. Un eje óptico perpendicular al plano de la imagen y al sensor.
Fuente: El autor.

2.8.2.2. Campo visual.

El campo visual se refiere al área visible perpendicular que circunda a un punto que se ha superpuesto al eje óptico, ya que esta área depende de la distancia a la que se encuentre dicho punto con relación al ojo o cámara, el campo visual queda definido por el ángulo medido entre el eje óptico y el punto visual mas lejano, dicho ángulo determina las áreas visuales a cualquier distancia, se ha calculado que en promedio los humanos tenemos un campo visual horizontal de 60 grados hacia la nariz y de 100 grados hacia afuera, en sentido vertical poseemos un campo visual de 60 grados hacia arriba y 70 grados hacia abajo, aproximadamente, las cámaras digitales tienen por regla general campos visuales de sección cuadrada y el ángulo en sentido vertical suele ser el mismo que el horizontal.

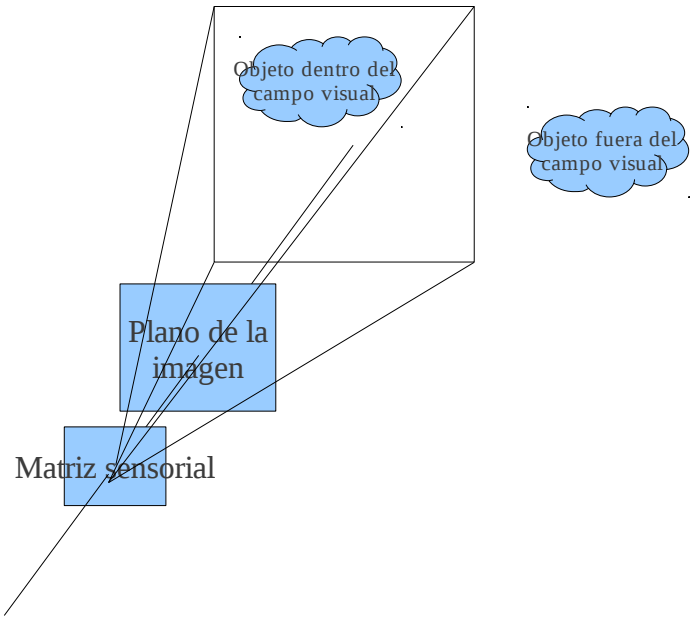

Figura 18. Representación del campo visual.

Fuente. El autor.

2.8.3. VISIÓN BINOCULAR.

Comprendemos como un sistema de visión binocular a aquel que utiliza dos ojos o cámaras para ver a un mismo objeto, la principal característica de este sistema es que permite la obtención muy exacta (dentro de un margen de distancias) de la profundidad a la que se encuentra un objeto que caiga dentro del campo visual estereoscópico, para la obtención de la profundidad mediante estereoscopía binocular no es necesario el conocimiento previo de las características del objeto en cuestión, si dicho objeto cae dentro del campo visual estereoscópico se podrá conocer su posición tridimensional. El uso de visión binocular disminuye el campo visual total que se obtendría si se usaran las cámaras en un sistema de visión mono-ocular, este hecho se deriva de que el campo visual estereoscópico es la intersección de los campos visuales individuales de cada cámara.

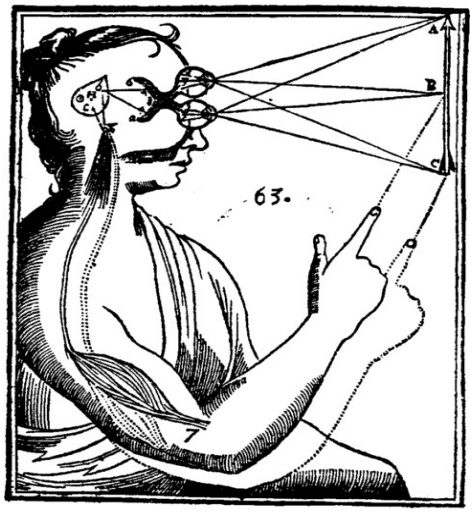

Figura 19. Diagrama de Descartes que ilustra la visión binocular.
Fuente: http://es.wikipedia.org/wiki/Archivo:Descartes_diagram.png

2.8.4. CAMPO VISUAL ESTEREOSCÓPICO Y CAMPO VISUAL RESIDUAL.

El campo visual estereoscópico se define como el área de intersección entre los campos visuales definidos por cada cámara en configuración de visión binocular, el campo residual es, para cada cámara, el campo o área visual no estereoscópica, es decir el campo visual que no interseca con el de la otra cámara.

2.8.5. DISTANCIA INTER-OCULAR.

En un sistema binocular, la distancia que separa a los puntos de intersección de los ejes ópticos de cada cámara con el sensor en sí (la retina en el caso biológico), se conoce como distancia inter-ocular, en los humanos es en promedio de 65mm.

2.8.6. ESTEREOPSIS, PARALAJE Y DISPARIDAD BINOCULAR.

El proceso de estereopsis se puede dar con un objeto que esté dentro del campo visual estereoscópico, consiste en la interrelación entre las imágenes de la misma escena capturadas por cada cámara (imágenes

estereoscópicas derecha e izquierda) con la finalidad de la obtención de la posición tridimensional de un objeto específico de la escena, a dicho proceso de interrelación se lo suele llamar también apareamiento estereoscópico.

Los mecanismos o procesos de estereopsis dependen del sistema estereoscópico usado, si es posible variar el ángulo medido entre los ejes ópticos la estereopsis se hace mediante mecanismos geométricos y de triangulación que se basan en el conocimiento de ángulo formado entre los ejes ópticos de cada cámara y los ángulos formados por los ejes ópticos y los planos de la imagen respectiva (paralajes), si los ejes ópticos son paralelos y fijos la estereopsis se hace mediante el cálculo de un valor llamado disparidad binocular, este valor se obtiene de la diferencia relativa entre las dos imágenes al ser captadas desde posiciones ligeramente distintas (distancia inter-ocular), la disparidad binocular es, al igual que el paralaje, dependiente y función de la profundidad.

2.8.7. LÍNEA BASE.

Es la línea que une a los centros de los receptores de las dos cámaras en configuración estereoscópica (CCD, Cmos, etc) u ojos (retinas).

2.8.8. APROXIMACIONES A LA ESTEREOSCOPÍA, EJES ÓPTICOS PARALELOS O CONVERGENTES.

Existen principalmente dos aproximaciones a la estereoscopía, aquellas en las que las cámaras varían el ángulo de convergencia de sus ejes ópticos con la finalidad de focalizar el objeto en cuestión en el centro de cada uno de los planos de la imagen (similar al sistema de visión humano) y, conocidos los ángulos de convergencia medidos desde la línea epi-polar (paralaje) y la distancia de la línea base, mediante triangulación, se obtiene una posición X,Y,Z del objeto con referencia a un sistema de coordenadas cuyo origen (o) usualmente se sitúa en medio de la línea base, la segunda

aproximación consta de dos cámaras y una disposición de ejes ópticos paralelos fijos y perpendiculares a la línea base, las dos cámaras se ubican separadas, la una respecto a la otra, una distancia constante y conocida denominada d (distancia de la línea base) o distancia inter-ocular, ambos modelos poseen determinadas características que los hacen mas recomendables en distintas situaciones, una explicación mas detallada se halla en:
VISION POR COMPUTADOR, Imágenes digitales y aplicaciones, Gonzalo Pajares, Jesús M. De la Cruz, capítulo 17.

La visión estereoscópica tiene una especial importancia en sistemas de robótica móvil basados en visión artificial, el objetivo de estos sistemas es lograr sistemas de estereoscopía artificial que permitan el desplazamiento de un robot sobre un terreno desconocido y en un ambiente no controlado sin marcas artificiales, la disponibilidad de cámaras para robótica entre las que se encuentran las AVRcam y las CMUcam es bastante limitada en el Ecuador y su importación resulta dificultosa, estas cámaras poseen características especiales como son el preprocesamiento y filtrado de la imagen, baja resolución, configuración y transmisión de la imagen por vía serial RS232 y, en algunos casos, la posibilidad de combinar dos cámaras y utilizar algoritmos que se incluyen con la compra para realizar sistemas de procesamiento simple como filtrados, detección de bordes, etc. En el caso de la CMUcam, estos algoritmos son privados, y su modificación o estudio se tornan imposibles o por lo menos ilegales, el tratamiento y procesamiento de imágenes digitales es una tarea que requiere, por lo general y dependiendo de la resolución de las cámaras, una enorme cantidad y tiempo de procesamiento, este problema se afronta comúnmente disminuyendo al mínimo posible la resolución de las cámaras usadas, con la correspondiente pérdida de detalle y, por lo tanto, de información que se causa, e incrementando al máximo la velocidad del procesador usado, a pesar de todo esto, los sistemas de estereoscopía suelen ser lentos, y para muchas aplicaciones la reacción de estos

sistemas no permitiría que sean considerados sistemas en tiempo real.

En contraposición, las cámaras de uso doméstico (webcams) presentan características que no se hallan comúnmente en las cámaras para robótica, como son la auto-calibración de brillo y contraste, y sobre todo, su costo es mucho menor, existen en una amplia variedad y son muy fáciles de hallar en el mercado, la única característica negativa que presentan, si se las quiere usar en un sistema de estereoscopía, es su elevada resolución comparada con las cámaras anteriormente mencionadas, casi siempre por encima del millón de pixeles, lo que hace que el tiempo de procesamiento requerido sea grande.

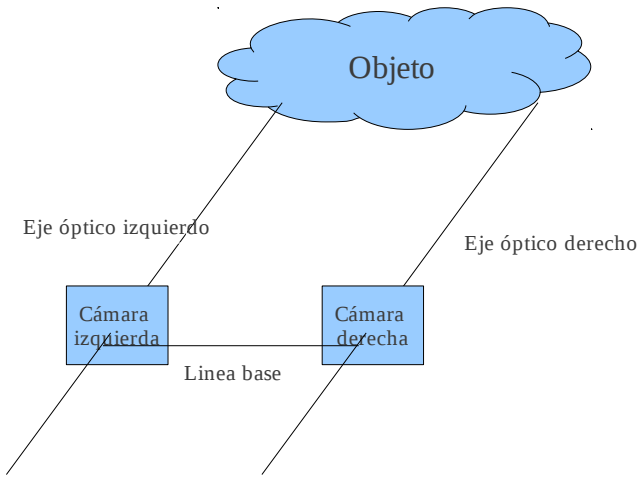

Figura 20. Estereoscopía de ejes ópticos paralelos, la profundidad se calcula mediante la disparidad binocular. Fuente: El autor.

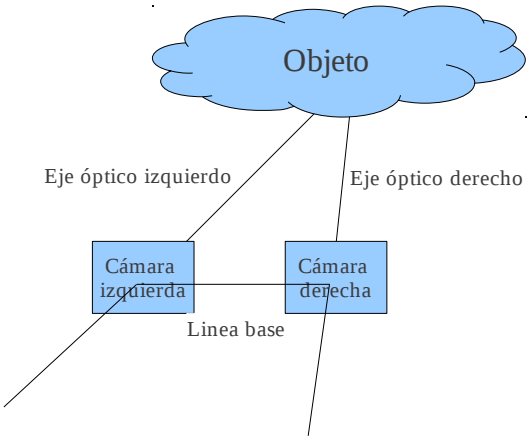

Figura 21. Estereoscopía de ejes ópticos convergentes, la profundidad se calcula mediante el paralaje.

Fuente: El autor.

CAPÍTULO 3. DESARROLLO.

3.1. REQUERIMIENTOS GENERALES DEL SISTEMA.

Considerando la información anteriormente presentada, se detallan a continuación los requerimientos generales del proyecto desarrollado:

El robot móvil desarrollado deberá disponer de:

- Ruedas en configuración diferencial.
- Sistema de control y posicionamiento de las ruedas basado en PWM.
- Sistema de visión estereoscópica de ejes ópticos paralelos.
- Sistema de posicionamiento absoluto basado en visión artificial estereoscópica y control difuso.
- Sistema de posicionamiento absoluto basado en sensores ultrasónicos de tiempo de vuelo.
- Sistema de posicionamiento relativo basado en odometría.
- Sistema de seguimiento de trayectorias geométricas básicas: trayectorias rectas, y giros sobre si mismo.

- Seguidor de línea por visión artificial y controlador difuso.

3.2. CONTROL DE MOTORES.

3.2.1. CONTROL DE SENTIDO DE GIRO Y VELOCIDAD DE MOTORES DC MEDIANTE PC.

El control de los dos motores DC a utilizar en la arquitectura diferencial del robot móvil en desarrollo implica el control tanto de velocidad, dirección de giro y posición de cada motor, el control deberá realizarse de manera completamente independiente para cada motor, tanto velocidad como sentido de giro y posición de cada motor serán controlados por dos sistemas iguales pero aislados el uno del otro, la velocidad de respuesta del sistema (tiempo desde que se envía una instrucción de cambio de sentido de giro o de velocidad hasta que efectivamente se realiza dicho cambio) deberá ser la menor posible, esto con la finalidad de obtener reacciones rápidas del sistema por ejemplo al evadir un obstáculo o al tomar una curva mientras se sigue una línea, además, debido al torque que son capaces de desarrollar los motores en cuestión, el control de velocidad deberá tener la posibilidad de realizarse de manera fina, es decir, de realizar cambios pequeños en la velocidad de giro, con esto se evitará que el móvil frene o acelere de manera brusca, produciendo posibles daños. Ya que de manera general el control del móvil será centralizado en la PC a bordo, las variaciones de velocidad y giro de ambas ruedas así como las lecturas de posición deberán hacerse desde la PC, se han seleccionado dos puertos seriales con esta finalidad, uno para cada motor, además, la cantidad de información enviada con el fin de controlar los motores deberá ser la mínima posible, pues el sistema de navegación se basa principalmente en el procesado de imágenes obtenidas desde cámaras web conectadas a la pc, cuando se utiliza un puerto USB

de manera frecuente, enviando y/o recibiendo información desde un puerto serial virtual, esto dificulta la comunicación del ordenador con la cámara, haciendo que la obtención de la imagen se torne lenta, por último, se deberá poder almacenar y mantener almacenada la velocidad y el sentido de giro en el que cada motor ha sido “seteado” por la PC, esto con la finalidad de posibilitar al procesador de la computadora el desarrollo de otras tareas mientras el móvil se encuentra desplazándose, un ejemplo de dichas tareas puede ser procesar las imágenes obtenidas desde las cámaras estereoscópicas.

3.2.2. CONTROL DE POTENCIA.

El controlador se realizó utilizando transistores mosfet (IRFZ44N – IRF9540) los cuales permiten el manejo de la potencia que los motores utilizados demandan, sin embargo el costo de cada mosfet es significativamente alto, al requerir de dos puentes H, uno para cada motor, resultó económicamente mas conveniente realizar el cambio de sentido de giro mediante dos relés y utilizar únicamente un transistor mosfet de canal N que genera los pulsos PWM para cada motor, el diagrama del circuito utilizado se muestra a continuación:

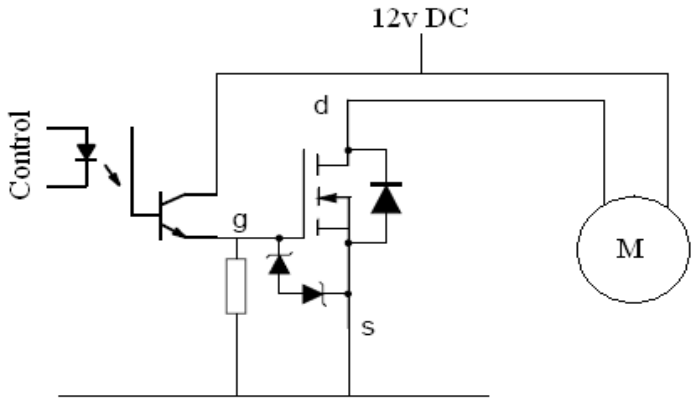

Figura 22. Control de potencia mediante mosfet.

Fuente: El autor.

La resistencia Pull Down en la compuerta es de 4.7K y el opto-acoplador un 4N25.

3.2.3. SISTEMA ELECTRÓNICO DE CONTROL DE VELOCIDAD.

Este sistema es el encargado de recibir, por vía serial, la velocidad y el sentido de giro y mantener al motor en dichas condiciones hasta recibir una nueva orden, se ha desarrollado utilizando dos microcontroladores pic 16F628 para cada motor, el objetivo es almacenar la velocidad y el sentido de giro utilizando el registro de memoria del puerto b del primer microcontrolador y detener la ejecución del programa de éste hasta que reciba un nuevo dato serial, mientras el programa en el segundo microcontrolador se mantiene en constante ejecución leyendo en su puerto b la velocidad almacenada en el puerto del primer pic, y generando pulsos con un ancho proporcional al valor leído en el puerto (PWM), de esta manera se hace mínimo el envío de información desde la PC, logrando que únicamente sea necesario enviar un dato serial cuando se desea cambiar la velocidad o el sentido, de esta forma se tiene una comunicación serial estándar RS232 entre la PC y el primer microcontrolador y una comunicación paralela entre los dos microcontroladores con esto se logra almacenar en el registro del puerto b la información sobre el sentido de giro y la velocidad a pesar de que la computadora no esté enviando información.

En la siguiente imagen se muestra el circuito utilizado:

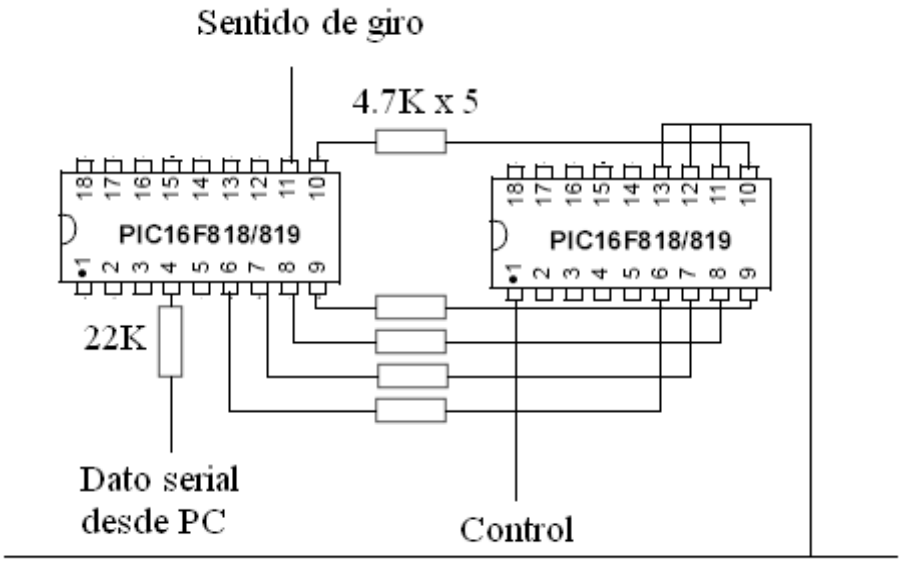

Figura 23. Circuito de control.
Fuente: El autor.

El microcontrolador de la izquierda es el encargado de esperar por un dato serial, y una vez recibido, almacenarlo en el puerto b, así, lo utilizamos a manera de “memoria”, el programa que ejecuta este microcontrolador es el siguiente:

```
define osc 20
trisb=0
include "modedefs.bas"
pausa var byte
portb=0

inicio:

   serin porta.5,N9600,pausa
   portb=(pausa-64)

goto inicio

end
```

Puesto que la instrucción serin detiene la ejecución del programa hasta que se reciba un dato serial, este programa estará detenido mientras no se desee cambiar la velocidad o el sentido de giro, como se ve, se han utilizado 5 bits para definir la velocidad (del B0 al B4), y el B5 se utiliza para definir el sentido de giro.

El segundo microcontrolador está continuamente leyendo el dato serial almacenado en los 5 bits menos significativos (los de menor valor) del puerto b del primer pic, y generando los pulsos necesarios para variar la velocidad del motor, el programa que corre en este microcontrolador es el siguiente:

```
define osc 20
trisb.0=1
trisb.1=1
trisb.2=1
trisb.3=1
trisb.4=1
trisb.5=1
trisb.6=1
trisb.7=1
trisa.2=0
pausa var word

inicio:
      pausa = portb
      pausa=pausa*50
      high porta.2
      pauseus pausa
      low porta.2
      pauseus (1550-pausa)'31*50=1550 para completar el ciclo de PWM
      '31 es la cosntante

while(portb=0)
      low porta.0
wend
goto inicio
end
```

Las pausas para la generación de la señal de control se han realizado en uS con la finalidad de evitar el golpeteo en los motores que se produjo al utilizar pausas en mS, como se ve, en el puerto a.2 se obtendrá una cadena de pulsos con una duración de 31 uS por cada pulso y 32 posibles combinaciones de alto y bajo, por ejemplo, si al puerto b llega el valor de 31 el bit a.2 estará idealmente de manera continua en estado de 1 lógico, si por el contrario se lee 0 en el puerto b, a.2 será 0v, para valores intermedios como 16, el puerto a.2 estará la mitad de tiempo en estado alto y la otra mitad en estado bajo, dando una velocidad de 50% al motor.

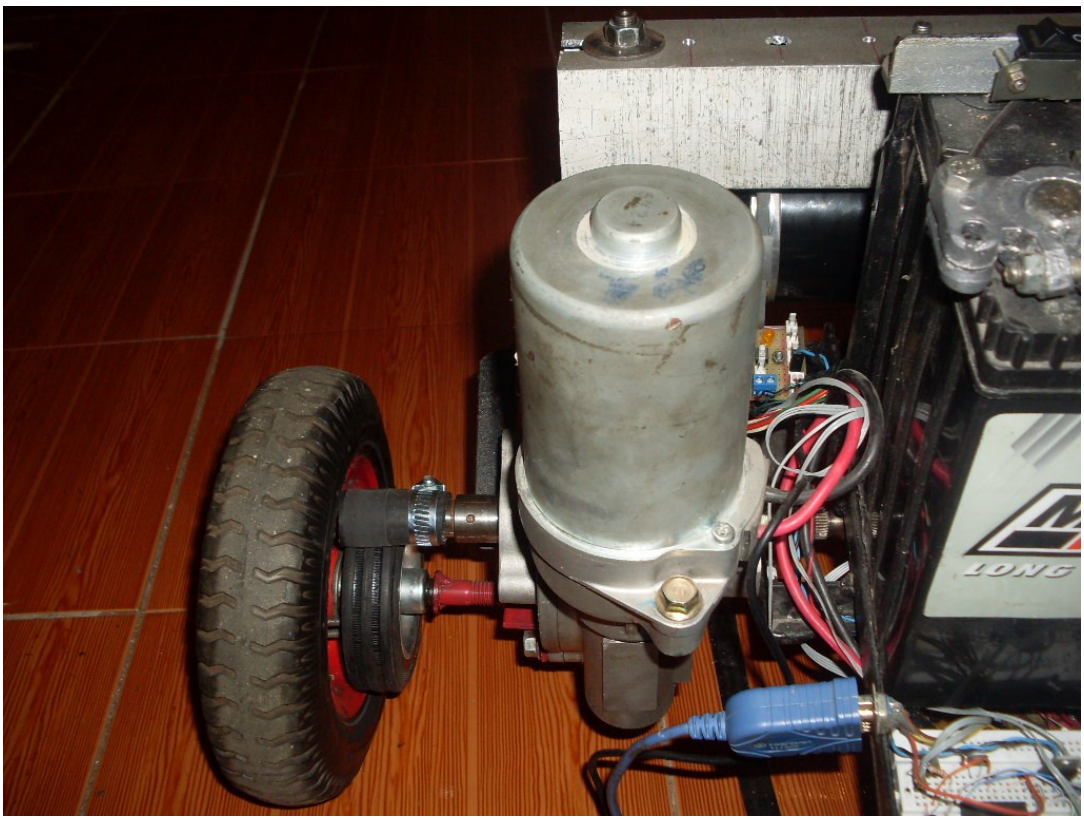

Imagen 1. El motor izquierdo acoplado a la rueda.

Fuente: El autor.

3.2.4. PROGRAMA DE CONTROL DE VELOCIDAD Y SENTIDO DE GIRO MEDIANTE PC.

Si utilizamos 5 bits para definir la velocidad del motor y el sexto para definir su sentido de giro (B.5 del primer pic), entonces tendremos 32 posibles velocidades, el software de control deberá escribir en el puerto serial adecuado un valor entre 0 y 31 para el control de velocidad y a dicho valor sumarle 64 para invertir el giro, puesto que el método serialPort .Write recibe como parámetro una cadena de caracteres, deberemos transformar el valor deseado de velocidad a su correspondiente carácter ASCCI para posteriormente escribirlo en el puerto, el programa de control utiliza una barra de desplazamiento (TrackBar) denominada velocidad_m1 para seleccionar la velocidad y sentido de giro, el cambio en el valor de el TrackBar ejecuta el método set_vel_m1, el cuál recibe un entero comprendido entre -31 que generará un PWM del 100% alto con un bit de sentido de giro igual a 1 ,y +31 que generará un PWM de 100% alto con bit de sentido de giro igual a 0, si se envía el valor de 0 el motor se detendrá correspondiendo con un PWM al 0% alto.

```
void set_vel_m1 (int velocidad){
        if ((velocidad >=0)&&(velocidad<=31)){
                String dato;
                dato=(Convert .ToChar(velocidad)).ToString();
                serialPort1 .Write (dato);
                velocidad_deseada =velocidad;
        }
        if ((velocidad <0)&&(velocidad>=-31)){
                velocidad_deseada =velocidad;
                String dato;
                velocidad =velocidad *-1;
                velocidad=velocidad + 32;
                dato=(Convert .ToChar(velocidad)).ToString();
                serialPort1 .Write (dato);
        }
}
```

El código anterior representa el método para setear la velocidad y el sentido de giro de las ruedas del robot.

Nótese que en caso de recibir un valor fuera de los parámetros establecidos, simplemente el método no hará nada.

3.2.5. REALIMENTACIÓN DE POSICIÓN ANGULAR Y VELOCIDAD ANGULAR MEDIANTE ODOMETRÍA.

La velocidad, al ser derivada de la posición, es posible obtenerla mediante software al conocer ésta, y puesto que uno de los principales objetivos es minimizar el envío y la recepción de información, el sistema electrónico será encargado únicamente de obtener las posiciones angulares de los ejes de los motores y transmitirla a la PC, para esto se han utilizado encoders con resolución de 4 grados, es decir que el microcontrolador podrá detectar un cambio de estado entregado por el circuito del encoder cada 4 grados de movimiento del eje del motor, con la finalidad de disponer de dos canales adicionales de comunicación a utilizarse para la obtención de distancias mediante sensores ultrasónicos de tiempo de vuelo, la información con respecto a la posición de los motores se obtendrá en la PC mediante el evento de comunicación serial DataReceived, de esta manera, cada vez que este evento se ejecute, el programa lo considerará como un cambio de estado en el codificador óptico acoplado al eje del motor, y en consecuencia como una variación de 4 grados en la posición angular de éste, el sentido de giro se presupondrá, dado que es el mismo programa el encargado de invertir la polarización de los motores, y puesto que el robot está diseñado para desplazarse sobre superficies planas horizontales se presupone que no existirán fuerzas externas que obliguen a los motores a girar sin estar polarizados; dejando así libre el byte enviado para la transmisión de cualquier tipo de información

adicional, de esta manera, el microcontrolador enviará un dato serial cualquiera cuando se produzca un cambio en el estado del codificador del motor, es importante notar que para que esta información sea receptada en la computadora debe existir movimiento en las ruedas, el circuito utilizado es el siguiente:

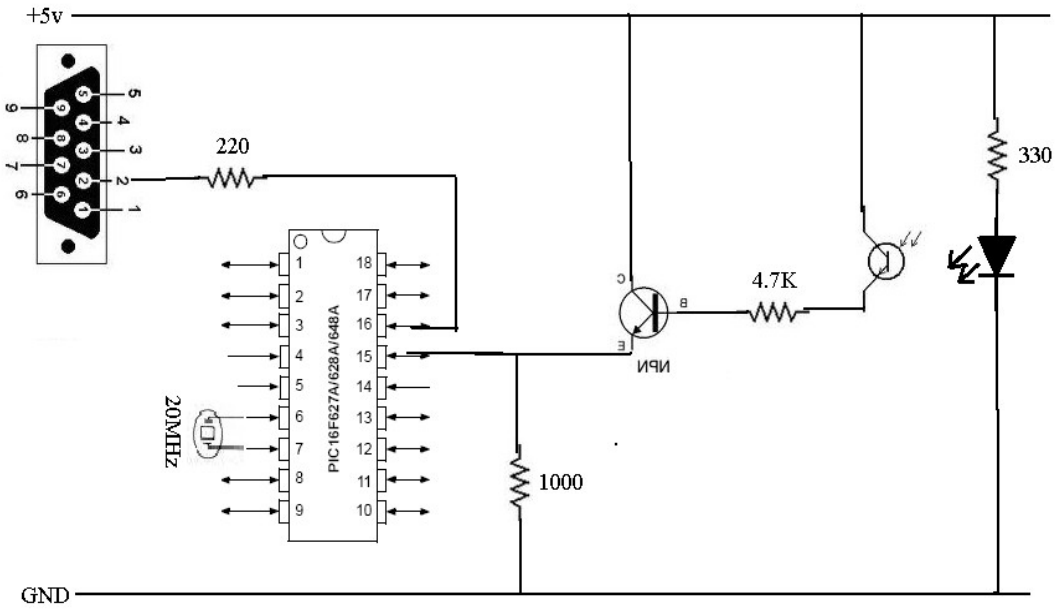

Figura 24. Circuito de odometría.

Fuente: El autor.

En la figura anterior se muestra el circuito de odometría, es posible utilizar los puertos libres del PIC para obtener información extra que se desee ingresar a la PC, en el presente proyecto se han usado para leer el valor entregado por los sensores ultrasónicos.

El programa en el microcontrolador es el siguiente:

```
define OSC 20
include "modedefs.bas"

trisb.0=1

inicio:

   while portb.0=1
    pauseus 500
wend
      serout portb.1,N9600,["1"]
      pauseus 50
   while portb.0=0
      pauseus 500
   wend
      serout portb.1,N9600,["1"]
      pauseus 50

goto inicio
end
```

En este caso se envía el dato arbitrario “1”, pudiendo ser éste cualquier dato obtenido por el PIC.
Las pausas agregadas de 500 y 50 microsegundos cumplen la función de pausas anti rebote, en el caso de que hayan oscilaciones o vibraciones en el disco perforado del sistema odométrico.

El método de dato recibido DataReceived ejecuta el siguiente código:

```
private void PuertoMotorDatoReceivido(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
{//CALCULO LA //POCISION ANGULAR
        if (velocidad_deseada>0){
                posicion_leida+=1;
        }
        if(velocidad_deseada<0){
                posicion_leida-=1;
        }
        posicion_angular_grados =posicion_leida*4; //SE MULTIPLICA POR 4 PUES ES LA
        //RESOLUCIÓN DEL CODIFICADOR
        //CONTROLO LA POCISON SI ES NECESARIO
        if(mover_por_angulo){
        if(sentido_aux){
                if(posicion_angular_grados>=(posicion_angular_deseada+posicion_angular_inicial)){
                                velocidad_deseada =0;
                                mover_por_angulo =false;
                }
        }
        else{
        if (posicion_angular_grados<=(-
        posicion_angular_deseada+posicion_angular_inicial)){
                                velocidad_deseada =0;
                                mover_por_angulo =false;
                        }
                }
        }
        contaux ++;
}
```

En donde la variable velocidad_deseada es global y obtiene su valor en el método set_vel_Motor, indica en consecuencia el porcentaje en alto de la señal PWM que se genera en ese instante y si el bit de sentido de giro está activo (<0) o inactivo (>0), las variables posición_leida y posición_angular_grados son también declaradas globales, esta última representa ya el ángulo que el eje del motor ha sido girado, pues se ha multiplicado ya por 4.

El método también se encarga de detener el motor una vez que el ángulo deseado se ha alcanzado, esto solamente si se ha llamado al método mover_motor, el cuál se encarga de llevar el eje del motor

hasta una posición angular deseada y a una determinada velocidad y sentido de giro:

```
public void mover_motor(int angulo,int velocidad,bool sentido){
        this .posicion_angular_deseada =angulo;
        if(!sentido) {
                velocidad_deseada =-velocidad;
                sentido_aux =false;
        }
        else {
                velocidad_deseada =velocidad;
                sentido_aux =true;
        }
        posicion_angular_inicial =posicion_angular_grados;
        mover_por_angulo =true;
}
```

Con la finalidad de poder incrementar o disminuir de manera sencilla la velocidad de cada motor, se han agregado los siguientes métodos:

```
public void incrementar_velocidad(int incremento){
        velocidad_deseada +=incremento ;
        set_vel_Motor (velocidad_deseada );
}
```

El método anterior recibe un valor entero positivo o negativo y hace el respectivo incremento de la velocidad del motor.

3.2.6. CÁLCULO DE LA VELOCIDAD DE LOS EJES DE LOS MOTORES.

Puesto que la velocidad de giro de las ruedas es necesaria para el seguimiento de una trayectoria definida por una de las fórmulas obtenidas en el modelamiento matemático de desplazamientos, se hace de vital importancia el diseño de un software que esté en capacidad de obtener la velocidad angular a partir de la diferencia en la posición angular leída después de un intervalo de tiempo dado.

Con la finalidad de obtener la velocidad de giro del eje del motor, el programa compara, cada 250 milisegundos (la frecuencia de ejecución del Timer), la posición angular leída con la posición angular leída anteriormente, y se calcula la velocidad en grados por segundo:

```
private void Calcular_controlar_velocidadTick(object sender, EventArgs e)
{
//CALCULO LA VELOCIDAD ANGULAR
if(velocidad_deseada!=0){
        velocidad_angular=(contaux*4)*(1000/Calcular_controlar_velocidad.Interval);
}
else {
        velocidad_angular=0;
}
}
```

La variable velocidad_deseada representa el valor entre 0 y 32 que se envía al setear el porcentaje de PWM, de esta manera el cálculo de la velocidad se hace solamente si el motor esta en movimiento, el Timer se ejecuta a intervalos de 250 milisegundos, la velocidad angular se multiplica por 4 para obtener el dato en grados por segundo.

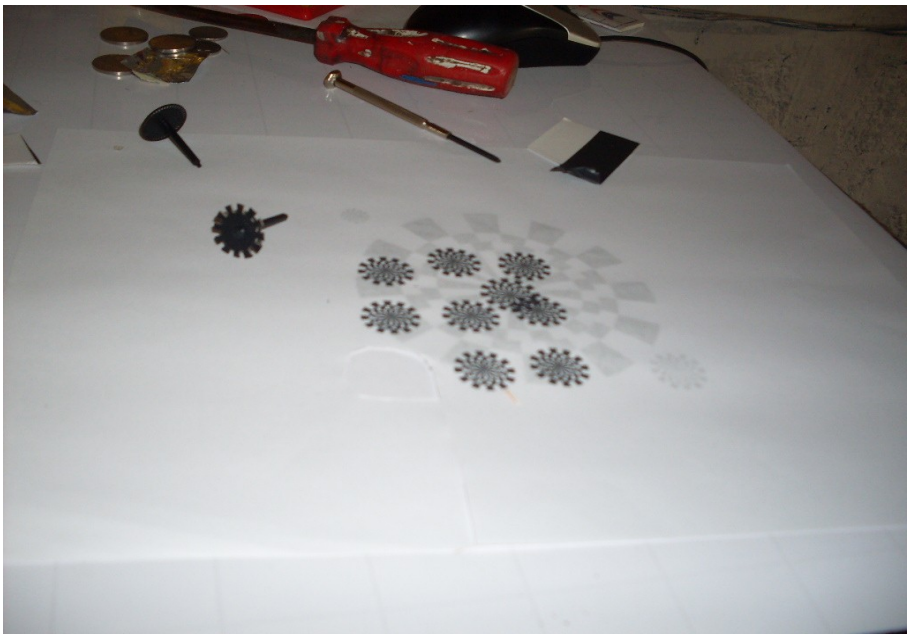

Imagen 2. Los encoders fueron impresos desde una computadora y luego fabricados en cartón.
Fuente. El autor.

Los métodos anteriormente descritos se han agrupado en la clase llamada Motor.cs:

Motor
+sentido:bool +mover_por_angulo:bool +posicion_angular_deseada:long +velocidad_angula:int +en_movimiento:bool +longitud:int
+Motor(puerto_motor:string):void +iniciar_puerto():void +setear_bit1(valor:bool):void +set_vel_motor(velocidad:int):void +mover_motor(angulo:int velocidad:int sentdo:bool):void +incrementar_velocidad(incremento:int):void

Figura 25. La clase Motor.cs.
Fuente: El autor.

3.3. DISEÑO DEL SISTEMA DE POSICIONAMIENTO POR SENSORES ULTRASÓNICOS.

3.3.1. LOS SENSORES ULTRASÓNICOS.

En el presente proyecto se han utilizado dos sensores ultrasónicos de tiempo de vuelo lvez4 los cuales tienen dos modos de funcionamiento, en el primer caso pueden entregar un ancho de pulso proporcional a la distancia que miden, en el segundo caso entregan un voltaje entre 0 y 5 voltios que representa la distancia medida, en ambos casos la proporción entre el valor entregado y la distancia leída por el sensor es lineal, y la fórmula para el cálculo de la distancia se encuentra en la hoja de datos del sensor.

3.3.2. LECTURA DE LOS SENSORES MEDIANTE LA PC.

Como se vio en el código del sistema odométrico, el dato enviado por el microcontrolador que lee los odómetros está libre para ingresar cualquier byte a la computadora, este byte se usará en esta parte del proyecto para ingresar el valor del tiempo de vuelo dado por el lvez4, puesto que el tiempo de cambio de estado del codificador óptico mientras giran las ruedas es muy corto, es decir, la frecuencia de los pulsos entregados por los odómetros es relativamente alta, teniendo cada pulso una duración aproximada de 200 uS, no se ha utilizado el modo de funcionamiento mediante ancho de pulso, sino que se usará el módulo ADC del PIC para leer el voltaje entregado por el lvez4, y este valor se enviará a la computadora por vía serial, para esto se ha modificado el código del microcontrolador de odometría, mostrado anteriormente, quedando de la siguiente manera:

```
define OSC 20 'se usa un oscilador externo de 20 MHz
define ADC_BITS 8 'tamaño de la muestra del ADC
DEFINE ADC_CLOCK 3 'reloj del ADC
DEFINE ADC_SAMPLEUS 50 'tiempo de muestreo del ADC
include "modedefs.bas"
trisa.0=1 'puerto conectado al sensor ultrasónico
trisb.0=1 'puerto conectado al odometro
adcon1=%00001110 ' configutación del ADC
distancia var byte

inicio:

   while portb.0=1
      adcin 0,distancia  'leo el lvez4
      pauseus 50  'pausa antirebotes
   wend
      serout portb.1,N9600,[distancia]  'envio el dato a la PC
      pauseus 50  'pausa antirebotes
   while portb.0=0
      adcin 0,distancia  'leo el lvez4
      pauseus 50  'pausa antirebotes
   wend
      serout portb.1,N9600,[distancia]  'envio el dato a la PC
      pauseus 50  'pausa antirebotes

goto inicio
end
```

Con esto hemos enviado los valores a la PC, resta únicamente leer los bytes recibidos agregando la siguiente línea de código al método Data_Received de cada puerto:

```
try{
        longitud = Encoding.ASCII.GetBytes(PuertoMotor .ReadExisting ())[0];
}
catch (Exception){}
```

En donde longitud es una variable pública global que almacena el valor obtenido por el ADC y que representa el tiempo de vuelo obtenido por el sensor ultrasónico.

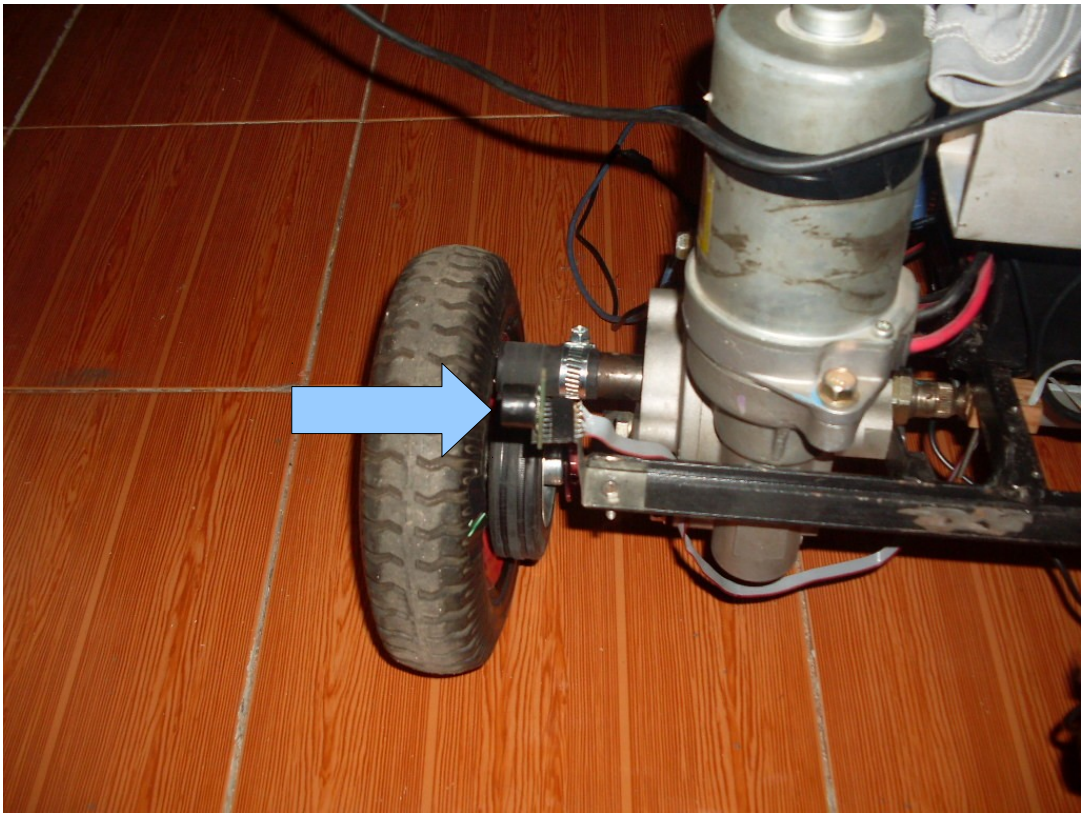

Imagen 3. Un sensor ultrasónico montado hacia el costado derecho del robot.

Fuente: El autor.

3.4. CONTROL DIFUSO.

Debido a las dimensiones del robot móvil que se desarrolla y a las dimensiones del área en la que deberá desplazarse, el control mediante un modelo matemático de dicho robot se tornó demasiado complejo, por ejemplo, considerar la inercia al momento de intentar seguir una trayectoria curva, determinar la velocidad de cada una de las ruedas con el objetivo de seguir la trayectoria dada, etc, son problemas que no se lograron resolver usando algoritmos de programación clásica, de esta manera se ha decidido realizar un sistema experto con motor de inferencia difuso, mediante el cuál se pretende realizar el control de una manera mas fina en cuanto a las variaciones de velocidad y de ángulo de dirección del móvil, lograr una

velocidad promedio aceptable al momento de seguir la trayectoria deseada, evitar abordar matemáticamente el problema y por último traducirlo a un lenguaje fácilmente comprensible y modificable dentro de la base de reglas del sistema.

La librería de clases Aforge, de código libre y disponible en Internet, es la herramienta seleccionada para el desarrollo del sistema, pues presenta características muy interesantes como son:

Es de código libre y abierto.

Presenta implementaciones de motores de inferencia difusos adecuados para el tipo de control que se desea realizar (fusificación por máximo y defusificación por centro de masa).

Se dispone de una amplia documentación y ejemplos dentro de la página web de Aforge: http://www.aforgenet.com/framework/ .

3.4.1. DETERMINACIÓN DE LAS VARIABLES DE ENTRADA.

Puesto que el objetivo del móvil es no perder la trayectoria que pretende seguir, la variable de entrada del sistema de control sería la desviación con respecto a dicha trayectoria, se ha planteado una sola variable de entrada, cuyo valor será obtenido mediante un algoritmo simple de procesamiento de imágenes programado en la computadora a bordo del móvil, la imagen será un vector unidimensional de pixeles en formato RGB de 1 x 200, de tal manera que si el robot está en la dirección correcta, el valor leído será igual a 100, si existe una desviación a uno de los dos lados, el valor leído será menor o mayor a 100, entre los límites de 0 a 200 fuera de los cuáles la imagen se pierde, la variable de entrada es, pues, un

número entre 0 y 200 que representa la desviación con respecto a la trayectoria que se pretende seguir.

Llamaremos a la variable de entrada “desviación”.

3.4.2. DETERMINACIÓN DE LAS VARIABLES DE SALIDA.

Las variables a controlar, ya que la arquitectura del móvil que se ha desarrollado es diferencial, serán necesariamente las velocidades de ambas ruedas, y la diferencia de dichas velocidades será comprendida como una corrección en el ángulo, es decir, si la variable de entrada está por debajo del valor deseado (100), se hará una corrección negativa, y por el contrario, si es mayor se hará una corrección positiva, variando la velocidad de ambas ruedas, las variables de salida serán dos números comprendidos entre 0 y 100, donde 0 representa un 0% de velocidad del motor (completamente detenido) y 100 representa el 100% que corresponde con la máxima velocidad.

Las variables de salida serán nombradas como “velocidadm1” y “velocidadm2” para el motor uno y el motor dos correspondientemente.

3.4.3. DETERMINACIÓN DE LOS CONJUNTOS DIFUSOS PARA CADA VARIABLE DE ENTRADA Y SALIDA.

Para la variable de entrada “desviación”, se han seleccionado los siguientes conjuntos difusos:

centrado

desviado_derecha
desviado_izquierda
poco_desviado_derecha
poco_desviado_izquierda
muy_desviado_derecha
muy_desviado_izquierda

De esta manera, el conjunto difuso Centrado corresponde a una desviación nula, es decir, cuando el robot está en su trayectoria correcta, la función de membresia de este conjunto deberá estar en torno a 100; los siguientes pares de conjuntos difusos corresponderán con funciones de membresia para valores de entrada mayores a 100 y menores a 100, a la derecha e izquierda respectivamente.

Para las variables de salida “velocidadm1” y “velocidadm2”, se han determinado los siguientes conjuntos difusos:

muy_rapido
rapido
poco_rapido
poco_lento
lento
muy_lento
detenido

3.4.4. DETERMINACIÓN DE LAS FUNCIONES DE MEMBRESIA PARA CADA CONJUNTO DIFUSO.

En lugar de funciones de membresia trapezoidales comunes se han seleccionado funciones de membresia triangulares, esto se debe a que la pertenencia de dichas funciones como completamente verdaderas existe únicamente en un punto, y las correcciones al momento de seguir la línea se hacen continuas, si se seleccionaran funciones trapezoidales, existirían valores en los cuales la corrección

del ángulo de dirección se tornaría constante a pesar de que existan variaciones en la desviación, lo cuál no sucede con las funciones triangulares.

Para los conjuntos difusos de la variable de entrada se han determinado las siguientes funciones de membresia:

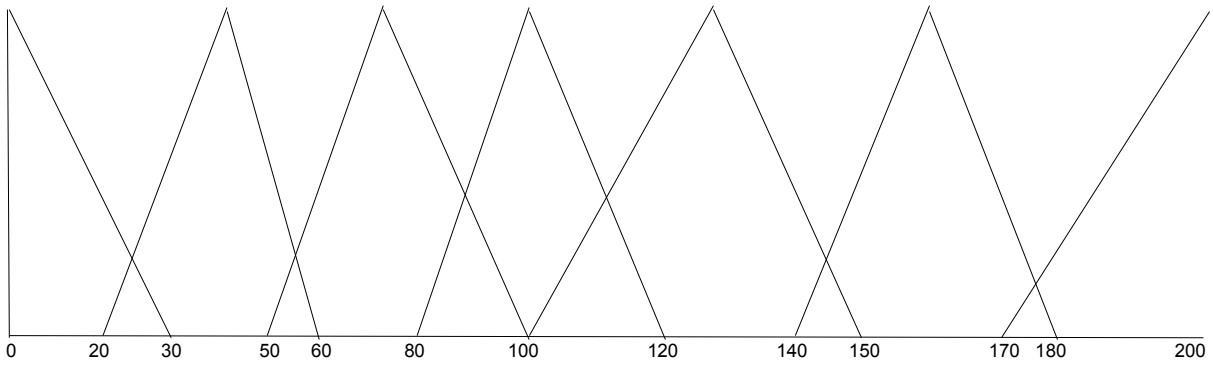

Figura 26. Conjuntos difusos de la variable de entrada.
Fuente: El autor.

Las funciones de membresia del gráfico anterior corresponden, de izquierda a derecha, con los conjuntos difusos:

muy_desviado_izquierda (M_D_I),desviado_izquierda (D_I), poco_desviado_izquierda (P_D_I), centrado (C_E), poco_desviado_derecha (P_D_D), desviado_derecha (D_D) y muy_desviado_derecha (M_D_D).

Para los conjuntos difusos correspondientes a las variables de salida se han seleccionado funciones de membresia trapezoidales puesto que con éstas se logra una mayor estabilidad obtenida del proceso de fuzificación, de esta manera se requiere una mayor variación en la variable de entrada para producir un cambio significativo en la velocidad de los motores, determinando las siguientes funciones de membresia:

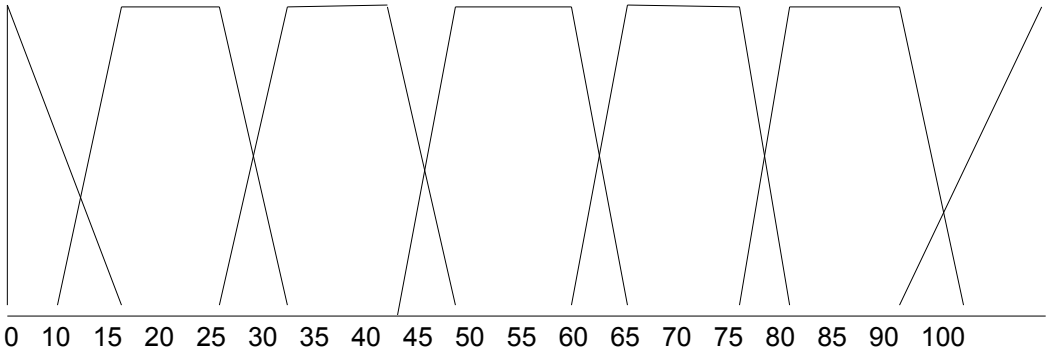

Figura 27. Conjuntos difusos para ambas variables de salida.

Fuente: El autor.

Donde cada función de membresia corresponde (de izquierda a derecha) con los conjuntos difusos:

Detenido (D_E), muy_lento (M_L), lento (L_E), poco lento (P_L), poco_rapido (P_R), rapido (R_A) y muy_rapido (M_R).

3.4.5. DETERMINACIÓN DEL TIPO DE MOTOR DE INFERENCIA A UTILIZAR.

De acuerdo a las recomendaciones presentadas en la bibliografía hallada se ha seleccionado un sistema de inferencia basado en lógica borrosa con desborrosificador por media de centros, e implicación borrosa por regla del mínimo, (Bonifacio Martín del Brío, 273).

Este tipo de sistema de inferencia es muy utilizado para el control de sistemas como el desarrollado en el presente trabajo, además es el sistema utilizado en la librería de clases Aforge.Fuzzy.

3.4.6. DETERMINACIÓN DE LA BASE DE REGLAS QUE DEFINIRÁN EL COMPORTAMIENTO DEL SISTEMA.

Mediante distintas pruebas con el sistema se ha obtenido el siguiente conjunto de reglas mediante las cuáles se ha llegado a un resultado satisfactorio en el comportamiento del robot:

IF Desviacion IS M_D_I THEN velocidadm1 IS D_E
IF Desviacion IS M_D_I THEN velocidadm2 IS M_R
IF Desviacion IS M_D_I AND Desviacion IS D_I THEN velocidadm1 IS M_L
IF Desviacion IS M_D_I AND Desviacion IS D_I THEN velocidadm2 IS M_R
IF Desviacion IS D_I THEN velocidadm1 IS L_E
IF Desviacion IS D_I THEN velocidadm2 IS M_R
IF Desviacion IS D_I AND Desviacion IS P_D_I THEN velocidadm1 IS P_L
IF Desviacion IS D_I AND Desviacion IS P_D_I THEN velocidadm2 IS M_R
IF Desviacion IS P_D_I THEN velocidadm1 IS P_R
IF Desviacion IS P_D_I THEN velocidadm2 IS M_R
IF Desviacion IS P_D_I AND Desviacion IS C_E THEN velocidadm1 IS R_A
IF Desviacion IS P_D_I AND Desviacion IS C_E THEN velocidadm2 IS M_R
IF Desviacion IS C_E THEN velocidadm1 IS M_R
IF Desviacion IS C_E THEN velocidadm2 IS M_R
IF Desviacion IS P_D_D AND Desviacion IS C_E THEN velocidadm2 IS R_A
IF Desviacion IS P_D_D AND Desviacion IS C_E THEN velocidadm1 IS M_R
IF Desviacion IS P_D_D THEN velocidadm2 IS P_R
IF Desviacion IS P_D_D THEN velocidadm1 IS M_R
IF Desviacion IS P_D_D AND Desviacion IS D_D THEN velocidadm2 IS P_L
IF Desviacion IS P_D_D AND Desviacion IS D_D THEN velocidadm1 IS M_R
IF Desviacion IS D_D THEN velocidadm2 IS L_E
IF Desviacion IS D_D THEN velocidadm1 IS M_R
IF Desviacion IS D_D AND Desviacion IS M_D_D THEN velocidadm2 IS M_L
IF Desviacion IS D_D AND Desviacion IS M_D_D THEN velocidadm1 IS M_R
IF Desviacion IS M_D_D THEN velocidadm2 IS D_E
IF Desviacion IS M_D_D THEN velocidadm1 IS M_R

3.4.7. PRINCIPALES CÓDIGOS DEL PROGRAMA DE CONTROL DIFUSO.

A continuación se presenta una síntesis del programa controlador difuso, así como una breve explicación de la función del código (en comentario).

Definición de variables.

```
//variable de entrada, desviacion con relación a la pocisión deseada
LinguisticVariable desviacion;
//conjuntos difusos pertenecen a la variable de entrada
FuzzySet centrado;
FuzzySet desviado_derecha;
FuzzySet desviado_izquierda;
FuzzySet poco_desviado_derecha;
FuzzySet poco_desviado_izquierda;
FuzzySet muy_desviado_derecha;
FuzzySet muy_desviado_izquierda;
//variables de salida, velocidad de los motores
LinguisticVariable velocidadm1;
LinguisticVariable velocidadm2;
//conjuntos difusos de las variables de salida
FuzzySet muy_rapido;
FuzzySet rapido;
FuzzySet poco_rapido;
FuzzySet poco_lento;
FuzzySet lento;
FuzzySet muy_lento;
FuzzySet detenido;
```

3.4.7.1. Definición de la base de datos para el almacenamiento de las reglas y del motor de inferenci.

```
Database fuzzyDB;//Defino la base de datos

InferenceSystem IS;//Defino el motor de inferencia.
```

3.4.7.3. Definición de las funciones de membresia de la variable de entrada.

```
this.muy_desviado_izquierda  =new FuzzySet ("M_D_I", new  TrapezoidalFunction(0, 30,
TrapezoidalFunction.EdgeType.Right ) );
this.desviado_izquierda  =new FuzzySet ("D_I", new  TrapezoidalFunction(20,40,60));
this.poco_desviado_izquierda  =new FuzzySet ("P_D_I", new  TrapezoidalFunction(50,70,100));
this.centrado  =new FuzzySet ("C_E", new TrapezoidalFunction(80,100,120));

this.poco_desviado_derecha  =new FuzzySet ("P_D_D",new   TrapezoidalFunction(100,130,150));
this.desviado_derecha  =new FuzzySet ("D_D",new   TrapezoidalFunction(140,160,180));
this.muy_desviado_derecha  =new FuzzySet ("M_D_D", new  TrapezoidalFunction(170,200,
TrapezoidalFunction.EdgeType.Left));
```

3.4.7.4. Definición de las funciones de membresia de las variables de salida.

```
detenido = new FuzzySet ("D_E",new TrapezoidalFunction(0, 15,
TrapezoidalFunction.EdgeType.Right ));
muy_lento=new FuzzySet ("M_L",new TrapezoidalFunction(10, 15, 25,30));
lento=new FuzzySet ("L_E",new TrapezoidalFunction(25, 30, 40, 45 ));
poco_lento=new FuzzySet ("P_L", new TrapezoidalFunction(40, 45, 55, 60));
poco_rapido=new FuzzySet ("P_R", new TrapezoidalFunction(55, 60,   70,75));
rapido =new FuzzySet ("R_A", new TrapezoidalFunction(70, 75, 85, 90 ));
muy_rapido  =new FuzzySet ("M_R",new TrapezoidalFunction(85, 100,
TrapezoidalFunction.EdgeType.Left ));
```

3.4.7.5. Definición las variables lingüísticas.

```
desviacion = new LinguisticVariable( "Desviacion", 0, 200 );
velocidadm1 = new LinguisticVariable( "velocidadm1",0,100 );
elocidadm2 = new LinguisticVariable( "velocidadm2",0,100 );

desviacion.AddLabel(muy_desviado_izquierda);
esviacion.AddLabel(desviado_izquierda );
desviacion.AddLabel(poco_desviado_izquierda);
desviacion.AddLabel(centrado );
desviacion.AddLabel(poco_desviado_derecha );
```

```
desviacion.AddLabel(desviado_derecha );
desviacion.AddLabel(muy_desviado_derecha );

velocidadm1 .AddLabel(muy_rapido);
velocidadm1 .AddLabel(rapido );
velocidadm1 .AddLabel(poco_rapido );
velocidadm1 .AddLabel(poco_lento );
velocidadm1 .AddLabel (lento );
velocidadm1 .AddLabel (muy_lento );
velocidadm1 .AddLabel (detenido);

velocidadm2 .AddLabel(muy_rapido);
velocidadm2 .AddLabel(rapido );
velocidadm2 .AddLabel(poco_rapido );
velocidadm2 .AddLabel(poco_lento );
velocidadm2 .AddLabel (lento );
velocidadm2 .AddLabel (muy_lento );
velocidadm2 .AddLabel (detenido);
```

3.4.7.6. Inicialización de la base de datos y del motor de inferncia.

```
fuzzyDB = new Database( );
fuzzyDB.AddVariable (this.desviacion );
fuzzyDB.AddVariable (this.velocidadm1 );
fuzzyDB.AddVariable (this.velocidadm2 );
//Inicializada la base de datos
IS = new InferenceSystem( fuzzyDB, new CentroidDefuzzifier( 10 ));
//Inicializado el motor de inferencia
```

3.4.7.8. Declaración de la base de reglas.

```
IS.NewRule( "Rule 1", "IF Desviacion IS M_D_I THEN velocidadm1 IS D_E" );
IS.NewRule( "Rule 2", "IF Desviacion IS M_D_I THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 3", "IF Desviacion IS M_D_I AND Desviacion IS D_I THEN velocidadm1 IS M_L" );
IS.NewRule( "Rule 4", "IF Desviacion IS M_D_I AND Desviacion IS D_I THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 5", "IF Desviacion IS D_I THEN velocidadm1 IS L_E" );
IS.NewRule( "Rule 6", "IF Desviacion IS D_I THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 7", "IF Desviacion IS D_I AND Desviacion IS P_D_I THEN velocidadm1 IS P_L" );
IS.NewRule( "Rule 8", "IF Desviacion IS D_I AND Desviacion IS P_D_I THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 9", "IF Desviacion IS P_D_I THEN velocidadm1 IS P_R" );
IS.NewRule( "Rule 10", "IF Desviacion IS P_D_I THEN velocidadm2 IS M_R" );
```

```
IS.NewRule( "Rule 11", "IF Desviacion IS P_D_I AND Desviacion IS C_E THEN velocidadm1 IS R_A" );
IS.NewRule( "Rule 12", "IF Desviacion IS P_D_I AND Desviacion IS C_E THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 13", "IF Desviacion IS C_E THEN velocidadm1 IS M_R" );
IS.NewRule( "Rule 14", "IF Desviacion IS C_E THEN velocidadm2 IS M_R" );
IS.NewRule( "Rule 15", "IF Desviacion IS P_D_D AND Desviacion IS C_E THEN velocidadm2 IS R_A");
IS.NewRule( "Rule 16", "IF Desviacion IS P_D_D AND Desviacion IS C_E THEN velocidadm1 IS M_R");
IS.NewRule( "Rule 17", "IF Desviacion IS P_D_D THEN velocidadm2 IS P_R" );
IS.NewRule( "Rule 18", "IF Desviacion IS P_D_D THEN velocidadm1 IS M_R" );
IS.NewRule( "Rule 19", "IF Desviacion IS P_D_D AND Desviacion IS D_D THEN velocidadm2 IS P_L");
IS.NewRule( "Rule 20", "IF Desviacion IS P_D_D AND Desviacion IS D_D THEN velocidadm1 IS M_R");
IS.NewRule( "Rule 21", "IF Desviacion IS D_D THEN velocidadm2 IS L_E" );
IS.NewRule( "Rule 22", "IF Desviacion IS D_D THEN velocidadm1 IS M_R" );
IS.NewRule( "Rule 23", "IF Desviacion IS D_D AND Desviacion IS M_D_D THEN velocidadm2 IS M_L");
IS.NewRule( "Rule 24", "IF Desviacion IS D_D AND Desviacion IS M_D_D THEN velocidadm1 IS M_R");
IS.NewRule( "Rule 25", "IF Desviacion IS M_D_D THEN velocidadm2 IS D_E" );
IS.NewRule( "Rule 26", "IF Desviacion IS M_D_D THEN velocidadm1 IS M_R" );
```

3.4.7.9. Ejecutar y probar el programa.

Para probar el correcto funcionamiento del programa se debe ingresar un valor de entrada (la desviación del robot) comprendida entre 0 y 200, en donde 100 significa que el robot está correctamente orientado hacia su posición objetivo, 0 esta totalmente desviado a la derecha y 200 totalmente desviado a la izquierda, estos valores pueden obtenerse mediante cualquier sensor como una cámara para seguir una línea o un sensor ultrasónico, al evaluar una entrada de 100:

```
IS.SetInput ("Desviacion",100);
```

Se obtendrá como salida el 100% de velocidad de ambos motores:

```
IS.Evaluate( "velocidadm1");

IS.Evaluate( "velocidadm2");
```

A los valores obtenidos se los transforma a los rangos de velocidad aceptados por los motores, esto se hace

multiplicándolos por 31 y dividiéndolos para 100, y se setea ambos motores usando el método set_vel:

```
set_velm1 (IS.Evaluate( "velocidadm1" )*31/100));

set_velm2 (IS.Evaluate( "velocidadm2" )*31/100));
```

Con esto, el robot hará las correcciones necesarias en la posición hasta encontrar su objetivo.

3.5. MODELO MATEMÁTICO.

Con la finalidad de analizar los desplazamientos del robot y calcular las trayectorias deseadas, se determinarán los siguientes puntos independientes dentro de la estructura del móvil:

$$Centro\,de\,guiado = Cg$$
$$Centro\,de\,la\,rueda\,izquierda = Ri$$
$$Centro\,de\,la\,rueda\,derecha = Rd$$

Y fuera de la estructura del robot:

$$Centro\,instantáneo\,de\,rotación = CIR$$

Nótese que el CIR solo existe en caso de que la velocidad lineal de la rueda 1 VlR1 ≠ VlR2.

Todas las ecuaciones se obtendrán para hallar el valor de la magnitud deseada en el centro de guiado del robot.

3.5.1. VELOCIDAD LINEAL DE LA RUEDA.

Las ruedas utilizadas quedan definidas mediante los siguientes parámetros:

$$Radio = r$$

$$Angulo\,de\,giro = \theta$$

$$Velocidad\,angular = \dot{\theta}$$

Es necesario también conocer la distancia entre los puntos de contacto de las dos ruedas con el piso:

$$Distancia\,entre\,puntos\,de\,contacto\,de\,las\,ruedas = D$$

La velocidad lineal del centro (eje) de la rueda en función de su velocidad angular queda determinada por:

$$Vl = \frac{\dot{\theta} * \pi}{180} * r$$

$$\dot{\theta} = \frac{Vl * 180}{r * \pi}$$

3.5.2. TRAYECTORIAS RECTAS.

Una trayectoria recta deseada se define por los siguientes parámetros:

$$Longitud\,de\,la\,trayectoria = \Delta d\,(m)$$

$$Velocidad\,lineal = Vl\,(m/s)$$

$$Dirección = Dir\,(+1 = adelante, -1 = atras)$$

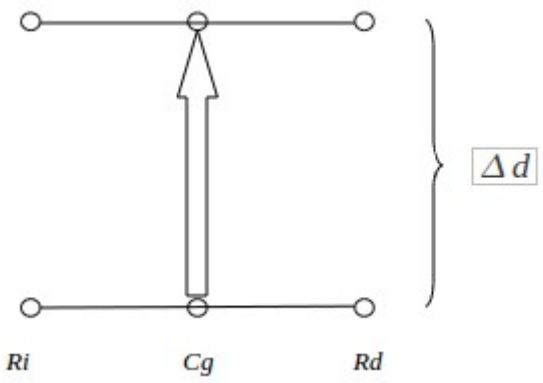

Figura 28. Una trayectoria recta.
Fuente: El autor.

Puesto que la trayectoria del Cg es en línea recta:

$$\Delta d\,Cg = \Delta d\,Ri = \Delta d\,Rd$$
$$VlCg = VlRi = VlRd$$

Sabemos que:

$$Vl = \frac{\Delta d}{t}$$

Donde:

$$Tiempo = t$$

La trayectoria recta quedaría definida por:

$$t = \frac{\Delta d}{VlCg}$$

Si se conoce la relación entre la velocidad lineal a la que se desplaza una

rueda y la velocidad angular a la que gira:

$$VlCg = \frac{\dot{\theta} * \pi * r}{180}$$

Donde r es el radio de la rueda.

Despejando la velocidad angular:

$$\dot{\theta} = \frac{VlCg}{\frac{\pi * r}{180}}$$

Lo que significa que si deseamos realizar un desplazamiento recto de longitud Δd a una velocidad lineal del robot $VlCg$ deberemos setear ambas ruedas, al mismo tiempo, a una velocidad angular $\dot{\theta} = \frac{VlCg}{\frac{\pi * r}{180}}$ durante un tiempo $t = \frac{\Delta d}{VlCg}$.

3.5.3. TRAYECTORIAS CIRCULARES.

Una trayectoria circular deseada queda definida por los siguientes parámetros:

$$Angulo\,de\,la\,trayectoria = \theta d\,(Grados)$$
$$Radio\,de\,la\,trayectoria = Rad\,(m)$$
$$Velocidad\,lineal = VlCg\,(m/s)$$

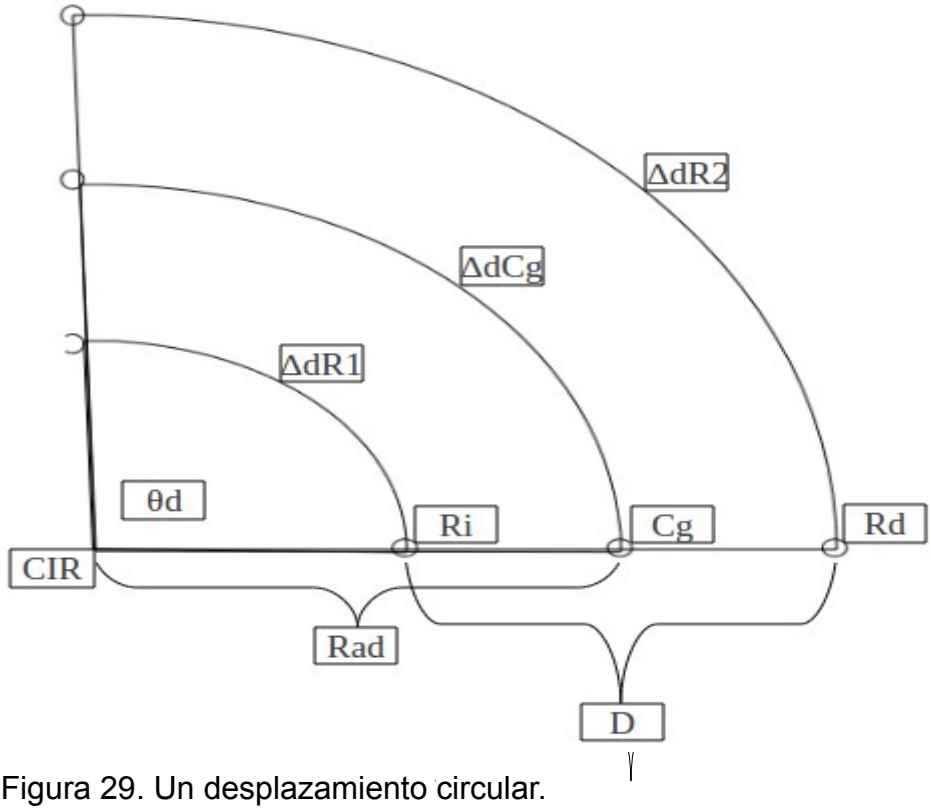

Figura 29. Un desplazamiento circular.

Fuente: El autor.

Se tiene que:

$$\Delta d\, Cg = \frac{\theta d * \pi}{180} * Rad$$

Entonces:

$$\Delta d\, R1 = (\frac{\theta d * \pi}{180}) * (Rad - \frac{D}{2})$$

$$\Delta d\, R1 = (\frac{\theta d * \pi}{180}) * (Rad + \frac{D}{2})$$

Tiempo:

$$VlCg = \frac{\Delta dCg}{t}$$

$$t = \frac{\Delta dCg}{VlCg}$$

$$t = \frac{(\frac{\theta d * \pi}{180}) * (Rad)}{VlCg}$$

Puesto que el tiempo es igual para los tres puntos en cuestión (Ri, Cg y Rd):

$$VlR1 = \frac{\Delta dR1}{t}$$

$$VlR2 = \frac{\Delta dR2}{t}$$

Se concluye que:

$$t = \frac{(\frac{\theta d * \pi}{180}) * (Rad)}{VlCg} = \frac{(\frac{\theta d * \pi}{180}) * (Rad - \frac{D}{2})}{VlR1} = \frac{(\frac{\theta d * \pi}{180}) * (Rad + \frac{D}{2})}{VlR2}$$

$$VlR1 = \frac{(\frac{\theta d * \pi}{180}) * (Rad - \frac{D}{2})}{\frac{(\frac{\theta d * \pi}{180}) * (Rad)}{VlCg}}$$

$$VlR2 = \frac{(\frac{\theta d * \pi}{180}) * (Rad + \frac{D}{2})}{\frac{(\frac{\theta d * \pi}{180}) * (Rad)}{VlCg}}$$

Entonces, si se requiere que el robot describa una trayectoria circular de radio Rad, de ángulo θd a una velocidad lineal del centro de guiado VlCg, se debe setear, al mismo tiempo ambas ruedas, a una velocidad lineal VlR1 y VlR2 respectivamente, si VlR1 ‹ VlR2 entonces la trayectoria será anti horaria, si VlR1 › VlR2 la trayectoria será horaria.

3.5.4. GIROS.

Un giro queda definido por dos parámetros:

$$Angulo\,de\,giro = \theta$$

$$Sentido\,de\,giro = horario\,u\,antihorario$$

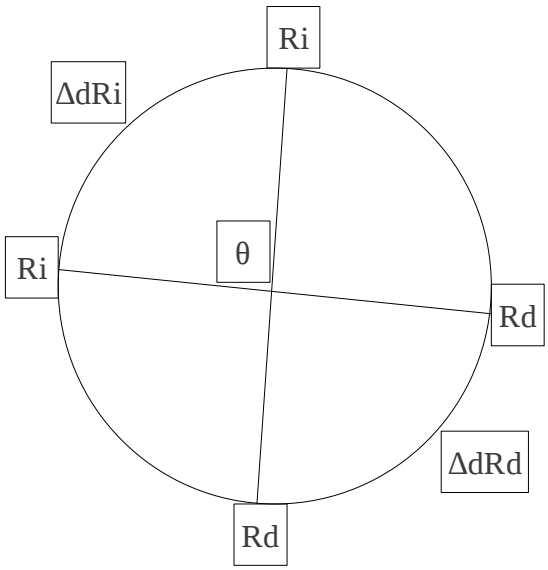

Figura 30. Un giro.

Fuente: El autor.

La distancia lineal recorrida por la rueda izquierda es igual a la recorrida por la rueda derecha:

$$\Delta dRd = \Delta dRi = \frac{\theta * \pi}{180} * \frac{D}{2}$$

Donde D es la distancia entre los puntos de contacto de las ruedas con el piso.

Conociendo la relación entre velocidad angular y lineal:

$$\dot{\theta} = \frac{Vl * 180}{r * \pi}$$

y el tiempo:

$$t = \frac{\Delta d Ri}{VlRi}$$

Donde VlRi es la velocidad lineal de la rueda izquierda y es igual a VlRd.

3.6. SISTEMA ESTEREOSCÓPICO BINOCULAR.

3.6.1. DESCRIPCIÓN.

Se construyó un sistema estereoscópico de ejes ópticos paralelos, esto debido principalmente a que el sistema de ejes ópticos convergentes presenta mayor complejidad en su realización mecánica y relativamente pocas ventajas, puesto que el robot enfrentará principalmente ambientes diseñados para el desenvolvimiento de seres humanos, y que se precisa una estereoscopía a corta distancia, se han montado las cámaras con una distancia inter-ocular igual a la distancia inter-ocular promedio humana, de 65mm, se han utilizado dos cámaras web iguales, con resolución de 0.1 mega pixeles, las imágenes capturadas son de 200x150 pixeles.

3.6.2. GEOMETRÍA DEL SISTEMA ESTEREOSCÓPICO.

A continuación se presenta una ilustración de la disposición física de los componentes del sistema estereoscópico binocular:

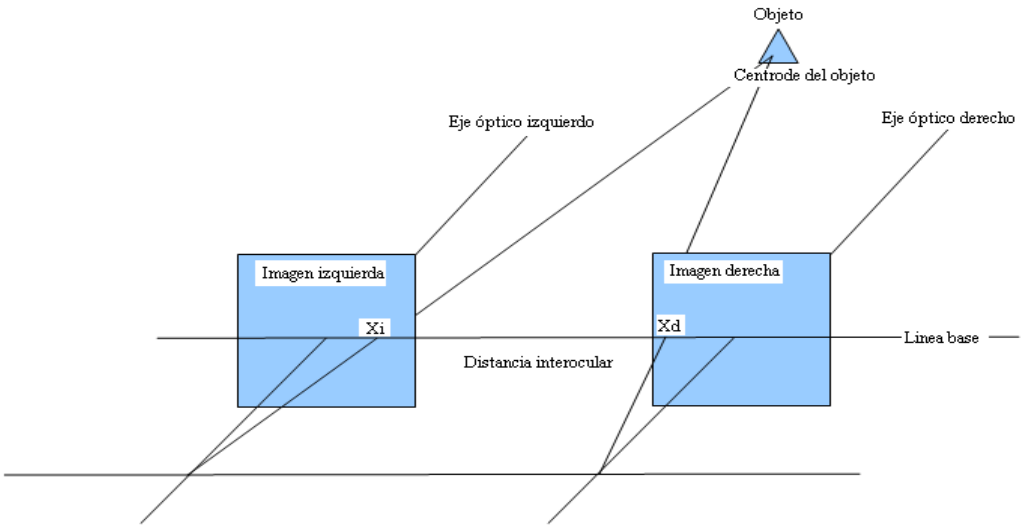

Figura 31. Un sistema estereoscópico binocular.

Fuente: El autor.

3.6.3. OBTENCIÓN DE LA DISPARIDAD BINOCULAR.

La disparidad binocular se calcula con relación a los centroides (Fig.) del objeto en cuestión (centroide del objeto en las imágenes izquierda y derecha), para esto se realiza un procesamiento de ambas imágenes que consiste en la siguientes etapas:

- Filtrado por color. Se filtra el objeto y se lo separa del resto de la imagen mediante el color de los pixeles que lo componen, para esto se usa la clase ColorFiltering de la librería de clases Aforge.Imaging.Filters, los límites para el filtrado de cada canal se establecen obteniendo un pixel de ejemplo dentro de la imagen del objeto en cuestión.
- Obtención de la escala de grises. Este es un requerimiento previo a la aplicación del filtrado de ruido, se realiza para dar a la imagen el formato de pixeles necesario para filtrar el ruido, se utiliza la clase Grayscale de la librería Aforge.Imaging.Filters.
- Erosión. En esta etapa se elimina posibles pixeles aislados u objetos pequeños que han superado el filtrado por color, considerados ruido, y que no son de interés, se utiliza la clase Erosion de la librería Aforge.Imaging.Filters.
- Dilatación. Con esto se pretende dar al objeto de interés un área, dentro de la imagen, similar al área original anterior a la erosión.
- Binarización y cálculo del centroide. Se binariza la imagen con un umbral predeterminado y al mismo tiempo se calcula el promedio de las posiciones de todos los pixeles correspondientes al objeto, se obtiene de esta manera el centroide necesario para el cálculo de la disparidad binocular.

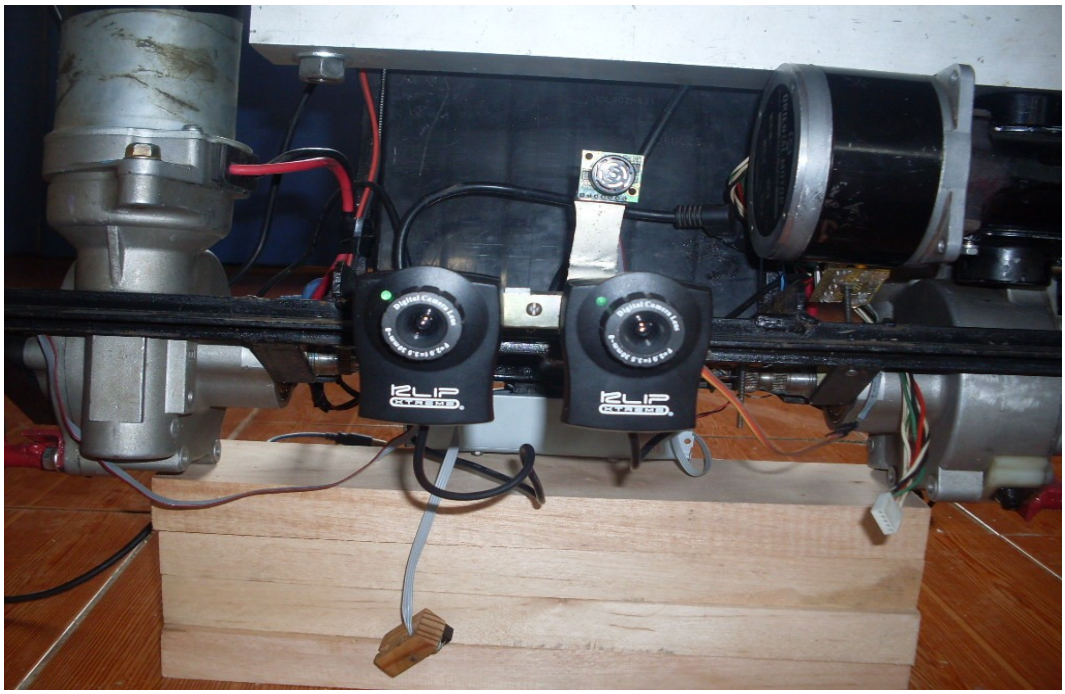

Imagen 4. Las cámaras en disposición binocular.

Fuente: El autor.

Una vez obtenidos los centroides del objeto tanto en la imagen izquierda como en la derecha, la disparidad binocular queda dada por el valor absoluto de la diferencia entre las posiciones en X (horizontales) de dichos centroides, de la siguiente manera:

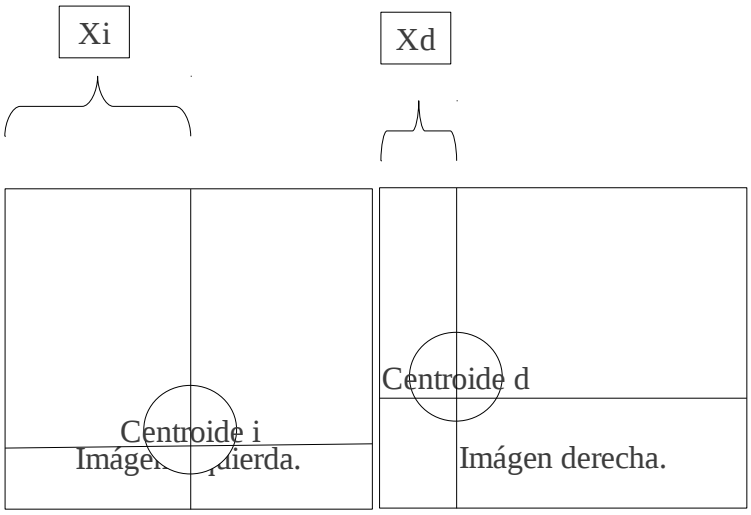

Figura 32. Cálculo de la disparidad binocular.

Fuente. El autor.

En la figura anterior se aprecia una representación de dos imágenes de un mismo objeto capturadas por dos cámaras en configuración estereoscópica, si llamamos Xi a la distancia en pixeles medida desde el borde de la imagen izquierda hasta el centroide del objeto en esta imagen, y Xd al mismo valor pero obtenido de la imagen derecha, la disparidad binocular queda dada por el valor absoluto de la diferencia entre Xi y Xd, en este caso la disparidad binocular esta dada en pixeles. La posición en Y de ambos centroides es irrelevante para el cálculo de la profundidad y de la disparidad binocular, sin embargo se debe procurar que sea la misma tanto en la imagen izquierda como en la derecha, esto con la finalidad de mantener al objeto de interés dentro del campo visual estereoscópico.

Recordando que la disparidad binocular guarda una relación matemática con la distancia entre el punto central de la línea base y, en este caso, el centro del objeto observado, queda ahora hallar dicha relación.

3.6.4. RELACIÓN MATEMÁTICA ENTRE PROFUNDIDAD Y DISPARIDAD BINOCULAR.

Si bien se podría pensar que la relación entre la disparidad binocular y la profundidad es lineal, esto no es así, el hecho mas notorio que se deriva de la relación no lineal entre estas dos magnitudes es que la percepción estereoscópica binocular disminuye con el aumento de la profundidad, de esta manera, si se varía en una unidad de distancia la profundidad desde la posición 0,0,0 del sistema de coordenadas estereoscópico (punto medio de la línea base) partiendo desde un punto cercano, se obtendrá una gran variación en la disparidad binocular, en cambio, si se varía esa misma unidad de distancia partiendo desde un punto lejano al origen de coordenadas, es probable que la disparidad binocular no varíe o lo haga mínimamente, cuan lejos sucede esto depende directamente de la resolución de las cámaras usadas, en el caso de la visión humana, la

percepción de la profundidad mediante estereoscopía binocular se pierde a partir de, en promedio, 16 metros, el sistema realizado utiliza cámaras de 0.1 megapixeles, experimentalmente se ha determinado que el sistema pierde la percepción de la profundidad a distancias mayores a 1,5 metros, si se reemplazan las cámaras por unas de 1 megamixel, se obtiene variaciones de la disparidad binocular hasta a una distancia de aproximadamente 3 metros, sin embargo, al elevar la resolución de las cámaras, el sistema se torna muy lento.

La función matemática que permite el cálculo de la profundidad a partir de la disparidad binocular se puede obtener mediante triangulación, (ver por ejemplo la Fig. Un sistema estereoscópico binocular) y por sistemas geométricos simples, sin embargo esto requiere el conocimiento previo y con exactitud de los parámetros físicos del sistema estereoscópico y de las características de las cámaras usadas, al ser cámaras de uso doméstico y al haber sido realizado el sistema de manera artesanal, dichos parámetros no han podido obtenerse.

Para obtener la función matemática que relaciona la profundidad y la disparidad binocular se han tomado datos experimentalmente para luego realizar un ajuste de curvas utilizando Microsof Excel, este método es mucho mas flexible que el anterior y presenta la ventaja de adaptarse a las condiciones y disposición geométrica de las cámaras, se han obtenido los siguientes valores:

Disparidad binocular (pixels)	Profundidad (cm)
71	30
56	40
41	50
31	60
23	70
16	80
12	90
8	100
6	110
3	120
2	130
1	140
0	150

Figura 33. Tabla de relación disparidad binocular – profundidad.
Fuente: El autor.

Como se ve en la tabla anterior, la percepción de la profundidad en el sistema desarrollado se pierde por sobre los 150 cm de distancia, pues la disparidad binocular se hace nula para todos los valores superiores a 1,5 metros, se hace evidente también que la función que las relaciona es de tipo no lineal e inversamente proporcional, en la siguiente figura se muestra la gráfica que relaciona las dos magnitudes:

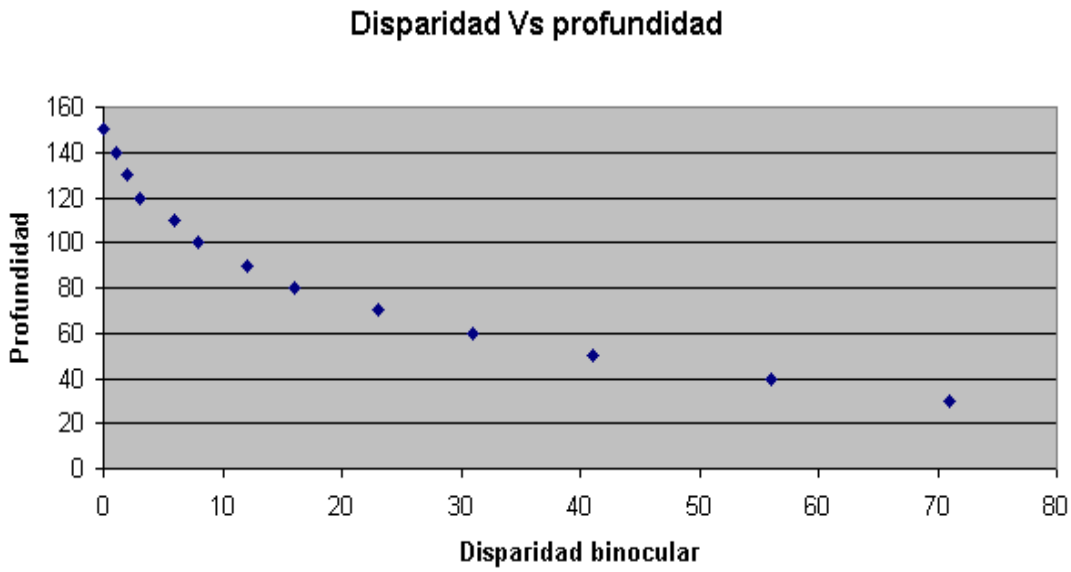

Figura 34. Gráfica de los datos obtenidos experimentalmente.

Fuente: El autor.

Se intentó realizar el ajuste de las curvas mediante funciones logarítmicas, exponenciales y polinomiales, obteniéndose el resultado mas fiel con esta última y de orden 3, la siguiente gráfica muestra la curva generada:

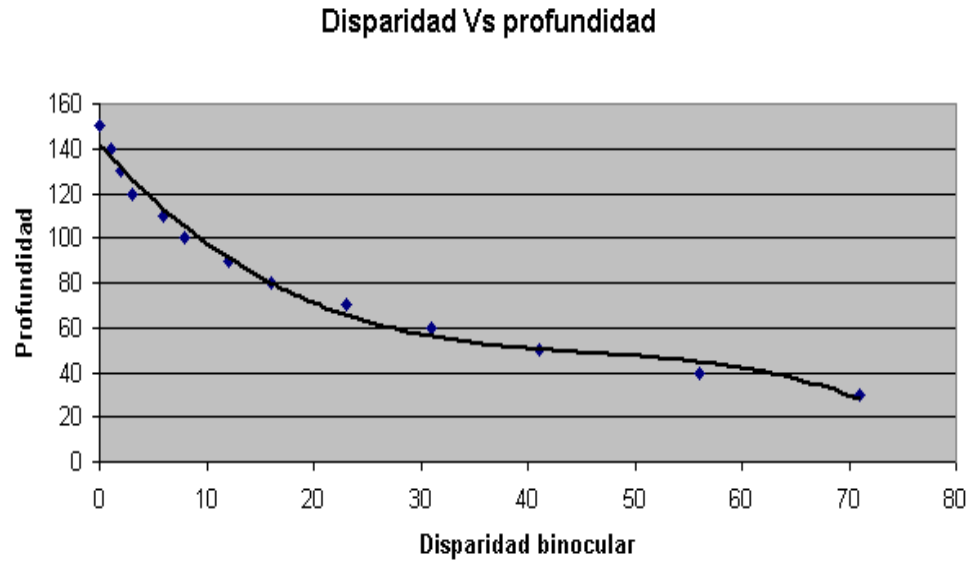

Figura 35. Curva ajustada mediante una función polinomial de tercer orden.

Fuente: El autor.

La función generada mediante Microsoft Excel es:

$$p = -0.0003 * d^3 + 0.0495 * d^2 - 3.2897 * d + 122,75$$

Donde p es la profundidad desde el centro de guiado del robot hasta el centroide del objeto y d es la disparidad binocular medida.

3.7. MODELAMIENTO POR SOFTWARE.

Una ves obtenidas las ecuaciones que describen una trayectoria dada, el siguiente paso es utilizar los métodos ya programados para el control de los motores con el fin de realzar un modelo software del robot, puesto que un robot diferencial se define mediante parámetros o características de su estructura física, se ha desarrollado el modelo software mediante la una clase denominada Robot_Diferencial, la clase posee los siguientes atributos públicos:

```
public int pos_X,pos_Y;//representan la posicion del robot en el mapa, valor en x e y respectivamente

public double angulo;// representa el  angulo del robot medido desde el eje x+ dentro del mapa

public  double D;// distancia entre los puntos de contacto de las dos ruedas motrices y el piso

private double relacion_transmicion_ruedas;// relacion de transmicion de los reductores de las
//ruedas

private double radio_ruedas;// radio de las ruedas medido en milimetros

private double diametro_ruedas;// diametro de las ruedas medido en milimetros
```

Con los parámetros anteriormente declarados es posible desarrollar los algoritmos de control para la descripción de trayectorias basándonos en las ecuaciones obtenidas en el modelamiento matemático.

El siguiente método realiza un desplazamiento en línea recta:

```
public void Desplazamiento_Recto(double velocidad_pwm, double longitud, bool sentido){

        double angulo_ruedas=(longitud*360)/(Math.PI*diametro_ruedas); //calculo el ángulo
        //que deben girar las ruedas para alcanzar la posición deseada

        double angulo_motores=angulo_ruedas*relacion_transmicion_ruedas; //calculo el
        //ángulo que deben girar las ruedas para alcanzar la posición deseada

        Motor_Izquierdo.mover_motor (Convert .ToInt16(angulo_motores),Convert.ToInt16
        (velocidad_pwm),sentido);

        Motor_Derecho.mover_motor (Convert .ToInt16(angulo_motores),Convert.ToInt16
        (velocidad_pwm),sentido);
        //Utilizo los métodos de control de motores para realizar el desplazamiento.

        if(sentido){
                pos_X+=Convert.ToInt16((Math.Cos(angulo*Math.PI/180))*(longitud));
                pos_Y+=Convert .ToInt16((Math.Sin(angulo*Math.PI/180))*(longitud));
                        }
        else{
                pos_X-=Convert .ToInt16((Math.Cos(angulo*Math.PI/180))*(longitud));
                pos_Y-=Convert .ToInt16((Math.Sin(angulo*Math.PI/180))*(longitud));
                }
        //Calculo la posición X,Y del robot después de realizar el movimiento solicitado
}
```

Como se ve, el método anterior usa llamadas al método de control de motores mover_motor, descrito anteriormente, con la finalidad de realizar el movimiento simultaneo de los dos motores.

El siguiente método realiza un giro sobre el centro de guiado:

```
public void Giro(double velocidad_pwm,double angulo, bool sentido){
        if(sentido){
                this.angulo+=angulo;
        }
        else{
                this.angulo-=angulo;
        }
        double longitud_arco=(angulo*Math.PI*D)/(360);//calculo la longitud del arco que
        //se requiere que cada rueda describa, no se refiere al ángulo de giro de la
        //rueda, sino a la longitud del desplazamiento.

        double angulo_ruedas=(longitud_arco*360)/(Math.PI*diametro_ruedas);//calculo el
        //ángulo que debe girar la rueda para describir un desplazamiento de la longitud
        //de arco previamente hallada

        double angulo_motores=angulo_ruedas*relacion_transmicion_ruedas;// calculo el
        //ángulo que deben girar los ejes de los motores

        Motor_Izquierdo.mover_motor (Convert .ToInt16(angulo_motores),Convert.ToInt16
        (velocidad_pwm),sentido);
        Motor_Derecho.mover_motor (Convert .ToInt16(angulo_motores),Convert.ToInt16
        (velocidad_pwm),!sentido);//Realizo el giro.
}
```

Puesto que el método mover_motor no considera la velocidad a la que se gira el motor, sino simplemente la posición a alcanzar, se observó que cuando se ejecuta un movimiento, uno de los dos motores se adelanta al otro, esto se da por las diferencias físicas entre ellos, como por ejemplo el rozamiento en las cajas reductoras, el siguiente código se ha incluido a un timer llamado Comparar_velocidades, el objetivo es comparar continuamente las posiciones de cada motor y saber si uno se adelanta a otro, en este caso se disminuye o aumenta convenientemente el voltaje (PWM) proporcionalmente al tiempo de diferencia en la posición de los motores, es decir, si se realiza un incremento y no se corrige el error, se realiza un incremento mayor, siendo éste un control proporcional:

```
private void Comparar_velocidadesTick(object sender, EventArgs e){
        if (Motor_Izquierdo .mover_por_angulo&&Motor_Derecho.mover_por_angulo){
        if(Motor_Izquierdo .velocidad_angular<Motor_Derecho.velocidad_angular){
        if ((Motor_Izquierdo .sentido==true && Motor_Derecho .sentido == true)||(Motor_Izquierdo
                .sentido==true && Motor_Derecho .sentido == false)){
        Motor_Izquierdo .incrementar_velocidad (1);}
        if ((Motor_Izquierdo .sentido==false && Motor_Derecho .sentido == false)||(Motor_Izquierdo
.sentido==false &&      Motor_Derecho .sentido == true)){
        Motor_Izquierdo .incrementar_velocidad (-1);
                        }}
        if(Motor_Izquierdo .velocidad_angular>Motor_Derecho.velocidad_angular){
        if ((Motor_Izquierdo .sentido==true && Motor_Derecho .sentido == true)||(Motor_Izquierdo
.sentido==true &&       Motor_Derecho .sentido == false)){
        Motor_Izquierdo .incrementar_velocidad (-1);
        }
        if ((Motor_Izquierdo .sentido==false && Motor_Derecho .sentido == false)||(Motor_Izquierdo
.sentido==false &&      Motor_Derecho .sentido == true)){
                Motor_Izquierdo .incrementar_velocidad (1);
        }}}
        }
```

Los métodos anteriormente descritos se han agrupado en la clase llamada Robot_diferencial:

Robot_diferencial
+posX:int +posY:int +angulo:double -motor_izquierdo:motor -motor_derecho:motor +D:double +relacion_transmicion_ruedas:double +radio_ruedas:double +diametro_ruedas:double
+desplazamiento_recto(velocidadPWM:double longitud_desplazamiento:double sentido:bool):void +giro(velocidadPWM:double angulo_de_giro:double sentido:bool):void

Figura 36. La clase Robot_diferencial.

Fuente: El autor.

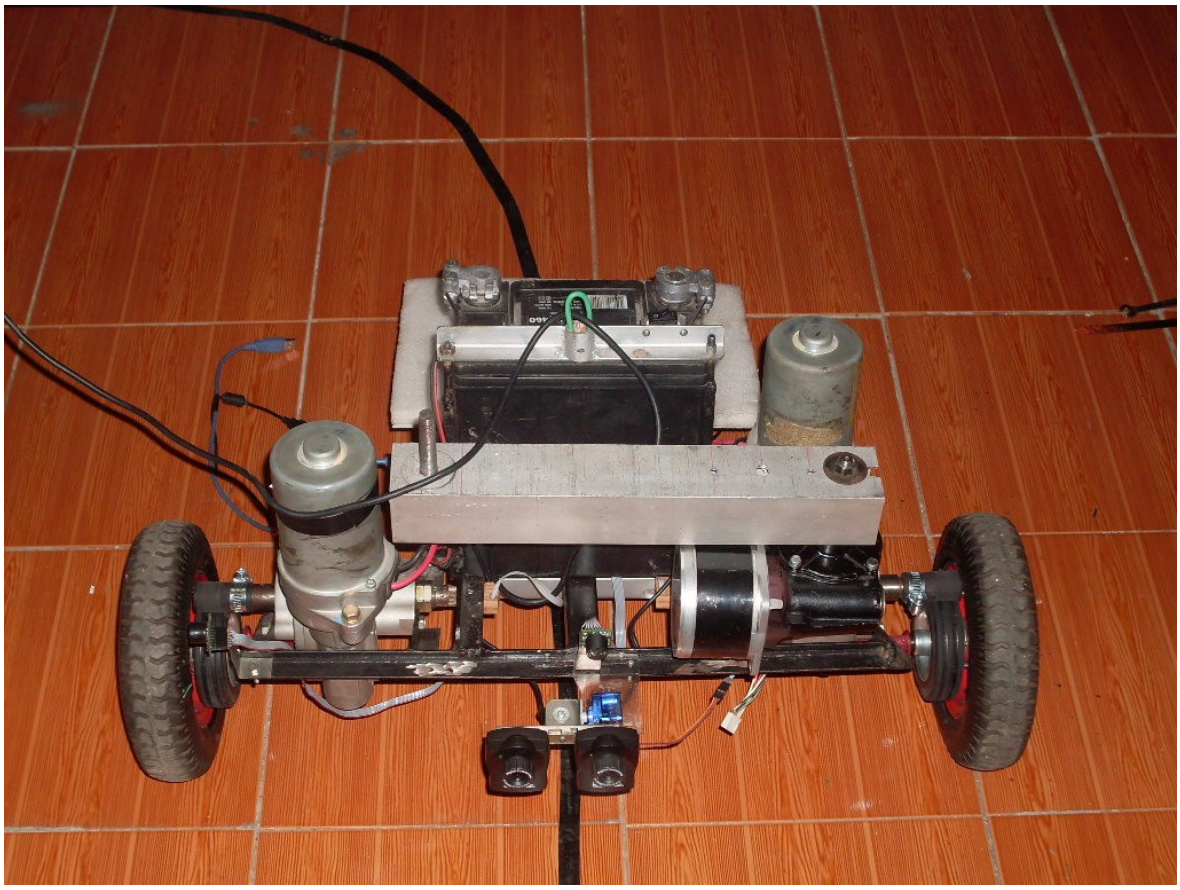

Imagen 5. El frente del móvil terminado.

Fuente: El autor.

CAPÍTULO 4. CONSTRUCCIÓN MECANICA.

4.1. MATERIALES.

4.1.1. MOTORES.

Para la construcción del robot móvil se han utilizado dos motores DC de dirección electrónica de auto, los motores, producidos por la empresa MANDO CORPORATION, poseen acoplada una caja reductora de tornillo

sin fin, puesto que los motores fueron adquiridos de un comerciante informal, ha sido imposible hallar datos suficientes para buscar sus especificaciones técnicas, se ha conseguido únicamente parte del número de serie del motor: 56300-1E503.

4.1.2. BATERIAS.

Como fuente de alimentación para los motores se ha utilizado una batería de plomo – ácido de marca Motorex y con número de serie NS40(S)460, la batería de auto posee las siguientes características:

V	12 V
CCA(22 grados C)	480 Amp
Capacidad de reserva	55 min
CCA(0 grados C)	380 Amp
CCA(0 grados F)	300 Amp

Para alimentar los circuitos de control del robot se ha utilizado una batería de plomo – ácido de menores dimensiones que la usada para alimentar a los motores, las principales características de esta batería se resumen a continuación:

V	12
CCA(22 grados C)	7 Amp

Esta batería, de marca Huanyn, posee como número de serie HYS1270.

4.1.3.RUEDAS.

Se utilizaron dos ruedas de goma inflables, las ruedas se han acoplado directamente al eje secundario de las cajas de reducción.

4.1.4.ESTRUCTURA.

La estructura del robot ha sido construida de hierro, usando ángulos de 3/4 x 1/8 y hierros T de 7/8 x 1/8, esta estructura mantiene juntos a los motores, las baterías y las ruedas, a continuación se presenta una tabla que detalla las dimensiones de el hierro usado.

Ángulos	Dimensiones			Sección	Peso	Valores estáticos		
	a	e	ex=ey	F	G	Jx=Jy	J1	J2
	mm	mm	mm	cm²	kg/m	cm³	cm³	cm³
5/8" x 1/8"	15,9	3,2	0.51	0.91	0.7	0.20	0.09	0.31
3/4" x 1/8"	19,1	3,2	0.58	1.11	0.9	0.37	0.17	0.57
7/8" x 1/8"	22,2	3,2	0.66	1.31	1.0	0.58	0.31	0.94

Los componentes de la estructura fueron soldados mediante suelda DC eléctrica con las dimensiones necesarias para sujetar los componentes antes mencionados, a continuación se presenta un esquema representativo del diseño hardware del robot desarrollado:

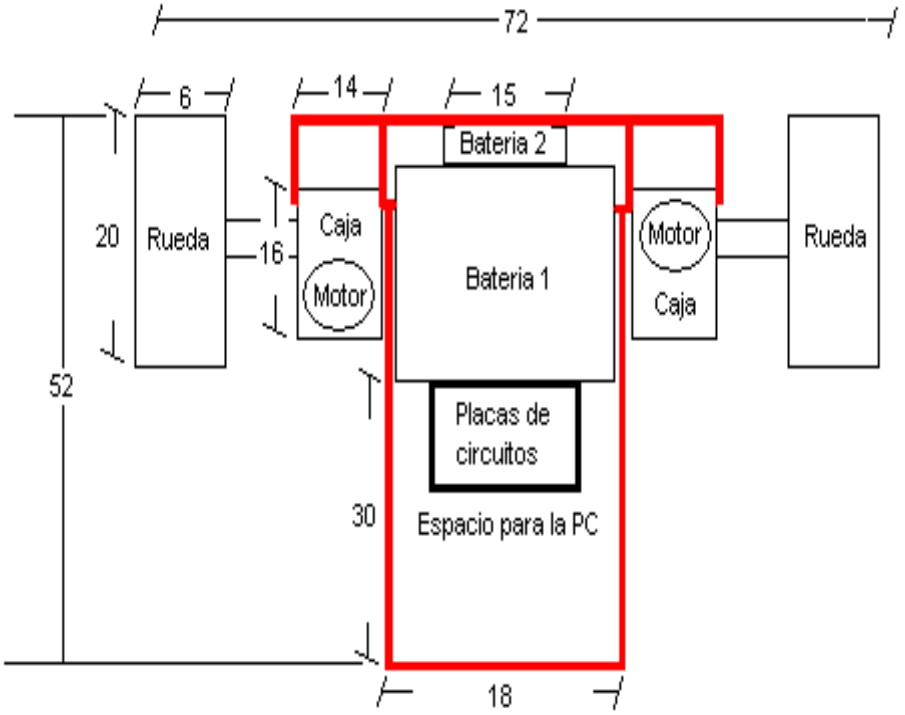

Figura 37. Dimensiones de la estructura mecánica.

Fuente: El autor.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES.

5.1. CONCLUSIONES.

El sistema de control y posicionamiento de motores funciona de manera estable y presenta reacciones en tiempo real, mediante el uso de los algoritmos diseñados se minimiza el envío y recepción de información mediante el bus USB, dejando libre el suficiente ancho de banda para el uso de las cámaras web.

El circuito de control desarrollado, mediante PWM, para el presente proyecto genera cadenas de pulsos correspondientes con el valor

seleccionado desde la computadora, la señal generada muestra cambios de estado casi instantáneos.

El modelamiento matemático y el modelo software para giros y desplazamientos se verificaron mediante pruebas de escritorio, y representan con exactitud el funcionamiento del robot, sin embargo la elevada masa total del robot y su correspondiente inercia hace que la descripción de trayectorias mediante el uso de las ecuaciones y sus programas no sea exacta, hecho que se suma al error provocado por la odometría.

El sistema odométrico permite estimaciones de la posición final después de realizar una secuencia de movimientos y posibilita giros de las ruedas del robot así como de los ejes de los motores dado un ángulo específico de giro.

Experimentalmente se determinó que el error aproximado en el uso de el sistema odométrico es de +- 10%, este error no es modelable y no se puede evitar sin modificaciones al hardware del robot.

El sistema estereoscópico presenta un funcionamiento aceptable dentro del rango de profundidad entre 10 y 100 cm, para profundidades mayores a los 100 cm se pierde la disparidad binocular debido a la baja resolución de las cámaras usadas, la disparidad binocular para objetos a mas de 100 cm es menor a un píxel, no siendo detectada, no se logró usar cámaras de mayor resolución puesto que el bus USB de la PC que controla el robot se satura.

Mediante las 26 reglas difusas y con la definición de las funciones de membresia presentadas se logró un comportamiento aceptable del

seguidor de línea con una velocidad promedio de 35 cm/s, acelerando en las trayectorias rectas y deteniéndose en las trayectorias curvas, así como un correcto posicionamiento relativo a un objeto mediante estereoscopía, las reacciones del sistema permiten considerar y modelar exitosamente, mediante la variación de la base de reglas del controlador difuso, modificaciones en el hardware del robot.

El sistema de lectura de sensores ultrasónicos presenta un funcionamiento adecuado, se logró eliminar por completo las mediciones falsas mediante los algoritmos de filtrado y mediante la sincronización en la emisión de pulsos ultrasónicos.

5.2. RECOMENDACIONES.

Considerando los siguientes hechos fundamentales, se formulan las posteriores recomendaciones:

Las características del proyecto han sido seleccionadas y diseñadas con la finalidad de hacer evidentes los problemas de investigación a los que se enfrenta un sistema robótico móvil, por ejemplo, la utilización de encoders y sistemas odométricos con mejores prestaciones que los usados en este robot, permite la descripción de trayectorias mucho mas fieles a las predichas por las ecuaciones del modelo matemático, de esta manera es posible disminuir considerablemente el error acumulado y estimar de mejor manera la ubicación final, sin embargo el problema principal que hace imposible la determinación exacta de la posición final mediante odometría estará siempre presente sin importar la calidad de los codificadores, de esta manera la necesidad de sistemas de posicionamiento absoluto se hace mas evidente y se estimula la investigación en esta área. De igual forma, se ha priorizado el diseño

del sistema general orientándolo hacia el uso de métodos de navegación mediante posicionamiento absoluto basado en marcas naturales, pues se considera que esta área de la robótica móvil presenta las mas prometedoras características en cuanto al impulso que puede dar al desarrollo de sistemas inteligentes, en consecuencia, se ha considerado que el desarrollo de la inteligencia artificial está muy ligado al estudio de los algoritmos de posicionamiento absoluto basados en marcas naturales, también se ha pretendido en el presente trabajo dar importancia a los de sistemas de control que no requieren un modelamiento matemático, como es el caso de la lógica difusa y los sistemas expertos basados en reglas, esto con la finalidad de dar una base hacia el uso de tales métodos de control aplicados a la robótica móvil, puesto que existen importantes limitaciones en el uso de matemáticas para modelar sistemas complejos como el entorno de desplazamiento de un robot móvil, se ha considerado importante incluir estos métodos de control al presente trabajo. Se seleccionó la visión estereoscópica como sistema central de adquisición de información para el posicionamiento absoluto del robot, si bien en la actualidad la visión artificial no es el método mas eficiente ni el mas fácil para el control de un robot móvil, la naturaleza ha mostrado que este es precisamente el método mas eficiente para la navegación en entornos desconocidos, y el desarrollo de algoritmos de visión artificial mono y binocular resulta el único camino completamente viable para el desarrollo de la movilidad en entornos desconocidos y dinámicos:

1. El desarrollo de robots móviles de uso industrial o doméstico debe basarse en sistemas de posicionamiento relativo y/o absoluto mediante marcas artificiales, puesto que de esta manera se obtendrán resultados prácticos y viables para su uso.

2. Es recomendable que la investigación en el área de robótica móvil se base y oriente hacia el diseño y uso de algoritmos de posicionamiento absoluto mediante marcas naturales, sin modificación o conocimiento previo del entorno.
3. En caso de usarse el presente proyecto como base para trabajos de investigación, los sistemas de posicionamiento relativo deben usarse únicamente como sistemas de apoyo a los de posicionamiento absoluto.
4. Se recomienda que el uso de visión artificial estereoscópica debe considerarse prioritario frente a otros métodos, como el de anillos de ultrasonido o escaners láser, estos últimos deben modelarse como sistemas de apoyo.
5. No es recomendable pretender abordar matemáticamente el modelamiento del comportamiento del robot al enfrentar un entorno desconocido, en lugar de esto, abordar el problema mediante técnicas flexibles e inexactas como la lógica difusa, los sistemas expertos o las redes neuronales artificiales.
6. Por último, se recomienda orientar el procesamiento de visión artificial hacia el uso de sistemas multiprocesador, esto con la finalidad de poder elevar la resolución de las cámaras usadas.

6. ANEXOS.

6.1. USO DE LAS CLASES, EJEMPLOS.

6.1.1. REQUERIMIENTOS DEL SISTEMA.

6.1.1.1. Compilador y S.O.

Todas las clases necesarias para controlar el robot se incluyen en el presente proyecto, para poder hacer uso de ellas es necesario un compilador de lenguaje c sharp para el

framework Microsoft.NET 2.0 o superior, si se desea controlar el robot desde un S.O. Linux se recomienda usar MonoDevelop, en el caso de Windows, es necesario instalar .Net Framework 2.0 o superior y se recomienda usar SharpDevelop 3.2 o superior o Microsoft VisualStudio.

6.1.1.2 Puertos seriales.

Como se ha explicado anteriormente, para poder controlar el robot se usa comunicación serial estándar RS232, no es necesario establecer las configuraciones de velocidad, paridad, etc. del puerto, puesto que éstas se configuran dentro del constructor de la clase Motor.cs.

Se requiere que la computadora posea mínimo dos puertos seriales, en el caso de usar conversores USB – RS232 es necesario verificar el nombre de los puertos virtuales creados e inicializar la clase Motor.cs con los puertos COM adecuados.

6.1.3. LIBRERÍAS DE CLASES Y CONTROLES.

Las librerías de clases y controles usados se deben agregar como referencias antes de compilar el código, todas las librerías necesarias se incluyen en el presente proyecto, se deben agregar referencias a las siguientes librerías:

Aforge.dll

AForge.Imaging.dll

Aforge.Fuzzy.dll

A demás, se requiere agregar el siguiente control:

WebCam_Capture.dll

6.2. INICIALIZACIÓN DE LAS CLASES.

A continuación se muestran códigos de ejemplo para la declaración e inicialización de las clases que controlan el robot:

6.2.1. LA CLASE MOTOR.CS.

Esta clase representa a los motores acoplados a las ruedas del robot.

Declaración de objetos.- Puesto que un robot diferencial esta compuesto por dos motores, es necesario declarar dos objetos de la clase motor, de esta manera:

```
Motor motor_izquierdo;
Motor motor_derecho;
```

Una vez declarados los dos objetos, es necesario inicializarlos con los parámetros adecuados:

```
motor_izquierdo=new Motor ("COM4");//Indico el puerto usado
motor_izquierdo .iniciar_puerto ();//Inicio el puerto
motor_derecho=new Motor ("COM3");
motor_derecho .iniciar_puerto ();
```

Métodos de la clase Motor.cs:

Método setear_bit1.- Se usa para setear el bit 6 del puerto b del microcontrolador memoria que almacena la velocidad de los motores, este bit puede ser usado con cualquier propósito, en este caso, el bit 6 del motor_izquierdo se usa para mover las cámaras estereoscópicas hacia arriba o hacia abajo:

```
motor_izquierdo.setear_bit1 (true);// sube las cámaras
```

Método mover_motor.- Se usa para girar el eje del motor un determinado ángulo, a una determinada velocidad y en sentido anti-horario u horario:

```
motor_izquierdo .mover_motor(1450,10,true); //1450 grados del motor corresponde con 360
//grados de la llanta
motor_derecho .mover_motor(360,10,true); //Mueve el eje del motor
//derecho 360 grados a una velocidad de 10, donde 30=100% de PWM, en
//sentdido horatio.
```

Método setear_pwm.- Envía indefinidamente al motor pulsos pwm con el porcentaje de alto deseado, 30 equivale al 100%, para invertir el sentido de giro es posible enviar al método un valor negativo:

```
motor_derecho.setear_pwm(10);//10 de 30 donde 30=100% de pwm

motor_izquierdo.setear_pwm(-20);//en sentido opuesto
```

Método incrementar_velocidad.- Produce un incremento del pwm generado en cada motor:

```
motor_izquierdo.incrementar_velocidad(12);// Incrementa 12 al
//pwm actual del motor.
```

Atributos públicos de Motor.cs.

La clase Motor.cs posee los siguientes atributos públicos:

```
public bool sentido;// representa el sentido de giro del motor, true=horario, false=antihorario
public bool mover_por_angulo;//indica si el motor ha alcanzado el angulo deseado mediante el metodo mover_motor (false) o si esta todabia en movimiento (true)
public long posicion_angular_grados;//LA POSICION ANGULAR EN GRADOS
public int velocidad_angular;//LA VELOCIDAD ANGULAR EN GRADOS POR SEGUNDO
public bool en_movimiento;//se usa para saber si las ruedas estan moviendose (true) o no (false);
public int longitud;//es el valor que entrega el sensor ultrasónico
// representa el tiempo de vuelo obtenido por cada sensor
//ultrasónico
```

6.2.2. LA CLASE ROBOT_DIFERENCIAL.CS.

Mediante la clase Robot_diferencial.cs se modela la estructura del robot, se declara e inicializa de la siguiente manera:

Declaración de objeto.- El siguiente código declara un objeto de la clase:

```
Robot_diferencial robot;
```

Inicialización de objeto.- Puesto que un robot diferencial esta compuesto por dos motores, y se conocen los parámetros necesarios para su modelamiento matemático, es decir, la distancia entre las ruedas motrices, el radio de las ruedas, la relación de transmisión de las cajas reductoras, y su posición inicial dentro de un plano de movimiento, el objeto recibe los parámetros mencionados:

robot=**newRobot_diferencial**(motor_izquierdo,motor_derecho,3.54166,105,675,-200,-300,0);//Robot diferencial compuesto por los dos motoresdeclarados anteriormente, con cajas reductoras de relación 3.54166, ruedas de 105 mm de diámetro, distancia entre ruedas motrices de 675 mm, y posición inicial X=-200, Y=-300 y ángulo = 0 grados.

Métodos de la clase Robot_diferencial.cs:

Método Desplazamiento_Recto.- Este método se usa para indicar al robot que debe realizar un desplazamiento recto de una longitud, velocidad y en un sentido dados, de la siguiente manera:

robot .Desplazamiento_Recto (10,500,true);//avanza 500 mm hacia adelante con pwm de 20

Método Giro.- Este método se usa para indicar al robot que debe realizar un giro de un determinado ángulo, a una velocidad deseada y en el sentido requerido:

robot .**Giro**(10,45,**true**);//giro de 54 grados horarios a pwm de 20

Métodos eliminar_movimientos, anadir_movimientos e iniciar_movimientos.- Estos métodos se usan para crear secuencias de movimientos, por ejemplo la siguiente secuencia:

```
robot .eliminar_movilientos ();
robot .anadir_movimientos ("Desplazamiento_Recto",vel,1000,true);
robot .anadir_movimientos ("Giro",vel,90,true);
robot .anadir_movimientos ("Desplazamiento_Recto",vel,500,true);
robot .anadir_movimientos ("Giro",vel,45,false);
robot .anadir_movimientos ("Desplazamiento_Recto",vel,300,true);
robot .anadir_movimientos ("Desplazamiento_Recto",vel,300,false);
robot .anadir_movimientos ("Giro",vel,45,true);
robot .anadir_movimientos ("Desplazamiento_Recto",vel,500,false);
robot .anadir_movimientos ("Giro",vel,90,false);
robot .anadir_movimientos ("Desplazamiento_Recto",vel,1000,false);
robot .iniciar_movimientos ();
```

Atributos públicos de la clase Robot_Diferencial.cs

La clase posee los siguientes atributos públicos:

```
public int pos_X,pos_Y;//representan la posición del robot en el
        //mapa poligonal

public double angulo;// representa el  ángulo del robot medido desde
//el eje x+
```

6.2.3. LA CLASE FUZZY.CS.

La clase Fuzzy inicializa sus atributos constantes, no son modificables desde fuera y el constructor no recibe parámetro alguno:

```
Fuzzy Proceso_Difuso;

Proceso_Difuso =new Fuzzy ();
```

6.2.POSIBLES APLICACIONES.

Debido a las características del presente proyecto, se recomiendan las siguientes aplicaciones posibles:

- Uso como base para el diseño y desarrollo de sistemas industriales.
- Uso como base para planteamiento de temas de investigación relacionados.

GLOSARIO DE TÉRMINOS.

Acelerómetro.- Sensor de aceleración.

Agente.- Todo sistema biológico o artificial que posee un objetivo, está en capacidad de obtener, mediante sistemas sensoriales, información de su entorno, y de ejercer, mediante sistemas motores, acciones sobre su habitad, con la finalidad de pretender su objetivo.

Cadena cinemática.- Un conjunto de varios elementos móviles, que se relacionan entre si mediante articulaciones, y en el que la posición de cada elemento depende de la posición del anterior.

Coeficiente de rozamiento.- Valor numérico que representa la fricción como oposición al desplazamiento relativo entre dos superficies en contacto.

Emisor infrarrojo.- En este caso, se refiere a la emisión infrarroja de señales digitales, con la finalidad de comunicación.

Encoder óptico.- O codificador óptico, sistema de medición de la posición angular que usa un disco entrecortado y un sistema de emisor-receptor lumínico para digitalizar los desplazamientos angulares del disco.

Entorno dinámico.- Cuando los objetos del entorno tienen velocidades relativas de magnitud relevante para el problema dado.

Error acumulable.- Cuando el error de una medida tomada mediante un sensor se suma al error de la medida siguiente, es un error acumulable.

Escáner láser.- Sensor usado en la construcción de modelos digitales de las formas geométricas del entorno, usando láser.

Eslabón.- En este caso, estructura física que relaciona, fija o une dos articulaciones de una cadena cinemática.

Estimación probabilística.- Método matemático probabilístico que se usa cuando no es posible hallar un modelo exacto de determinación de una magnitud, se obtiene un valor probable con un rango de error posible.

Fuerza centrífuga, centrípeta.- Fuerza opuesta a la centrípeta en un movimiento circular.

Fuerza de Coriolis.- Efecto de fuerza que experimenta un sistema en movimiento dentro de un sistema en rotación o giratorio.

Mapa poligonal.- Un mapa que representa a los objetos del entorno mediante polígonos regulares cerrados.

Mapa poligonal paramétrico.- Un mapa que representa a los objetos del entorno mediante polígonos regulares cerrados y los define paramétricamente, cada parámetro representa uno de los lados del polígono mediante las coordenadas de sus puntos inicial y final, considerando que se debe tratar de un polígono cerrado, así cada polígono paramétrico se define mediante un conjunto de pares de coordenadas.

Modelo matemático.- Una descripción matemática o un conjunto de relaciones matemáticas que modela y predice con exactitud el comportamiento de un sistema dado.

Modelo predictivo.- Un modelo matemático que permite estimar predictivamente el comportamiento de un sistema, previo a la ejecución del experimento, permite antelar el valor que tomará una variable considerando la situación actual de otras relacionadas, y tomar decisiones o controlar variables en base a dicha predicción.

Mono-ocular.- Cuando se captura una sola imagen de una escena, contrario a binocular: cuando se captura dos imágenes de la misma escena desde posiciones distintas.

Omnidireccionalidad.- Condición de un robot móvil que indica la independencia entre la orientación angular del robot y su posibilidad de desplazamiento, posibilidad de desplazarse en cualquier dirección sin necesidad de variar la orientación angular.

Percepción.- Sensación procesada, interpretación que se hace de la sensación, relacionándola con las condiciones y los objetivos del sistema.

Polígono regular.- Una forma bidimensional construida con lineas rectas sucesivas.

Polígono regular cerrado.- Un polígono regular en el que la última línea está conectada con la primera.

Potenciómetro.- Divisor de voltaje variable o resistencia variable.

PWM.- Modulación por ancho de pulsos.

Reconocimiento de patrones.- Obtención de descripciones características, típicamente valores numéricos, de un objeto, que lo hacen perteneciente a una clase dada de objetos y permiten su

posterior reconocimiento como un objeto de cierta clase.

Sensación.- Dato obtenido directamente de un dispositivo sensor, magnitud numérica que entrega un transductor de entrada y a la que no se le ha aplicado procesamiento informático.

Tiempo real.- Cuando un sistema cumple con las restricciones temporales que el entorno y el objetivo que pretende le imponen, es en tiempo real; es decir, cuando la reacción del sistema se da efectivamente dentro del tiempo requerido.

Torque.- En una palanca o una rueda, la fuerza multiplicada por la distancia.

Ultrasonido.- Sonido de frecuencia mayor a la máxima audible por el humano, es decir, superior a los 20000 Hz.

Visión artificial.- Visión mediante computadora y cámaras que pretende, mediante procesamiento de las imágenes digitales, la obtención de información y la interpretación de una escena dada.

BIBLIOGRAFIA.

Bonifacio Martín del Brío; Redes Neuronales y Sistemas Difusos; Segunda edición ampliada y revisada.

Stuart Russell, Peter Notving; Inteligencia Artificial Un Enfoque Moderno; Segunda edición.

Carlos A. Reyes; Microcontroladores PIC; Segunda edición.

Gonzalo Pajares Martisanz; Visión por computador Imágenes digitales y aplicaciones.

William Siler; Fuzzy Expert Systems and Fuzzy Reasonning.

Thomas Bräunl; EMBEDDED ROBOTICS Mobile Robot Design and Applications with Embedded Systems; Second Edition.

McGraw - Hill –
PIC.Robotics.A.Beginners.Guide.to.Robotics.Projects.Using.the.PIC.Micro

Anibal Ollero Baturone, Rbótica Manipuladores y Móviles.

Internet:

http://www.scielo.cl/scielo.php?pid=S0718-07642006000500003&script=sci_arttext

http://www.geintra-uah.org/system/files/TrabajoTuteladoMartaMarronRomera02.pdf

http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/15018191-196.pdf